# 68' MICRO JOURNAL

Singapore S $ 9.45 Hong Kong H $23.50
Malaysia M $ 9.45 Sweden 30.-SEK

**$2.95USA**

**OS-9 Atari Amiga Mac S-50**

**6800 6809 09008 68000 68010 68020 68030**

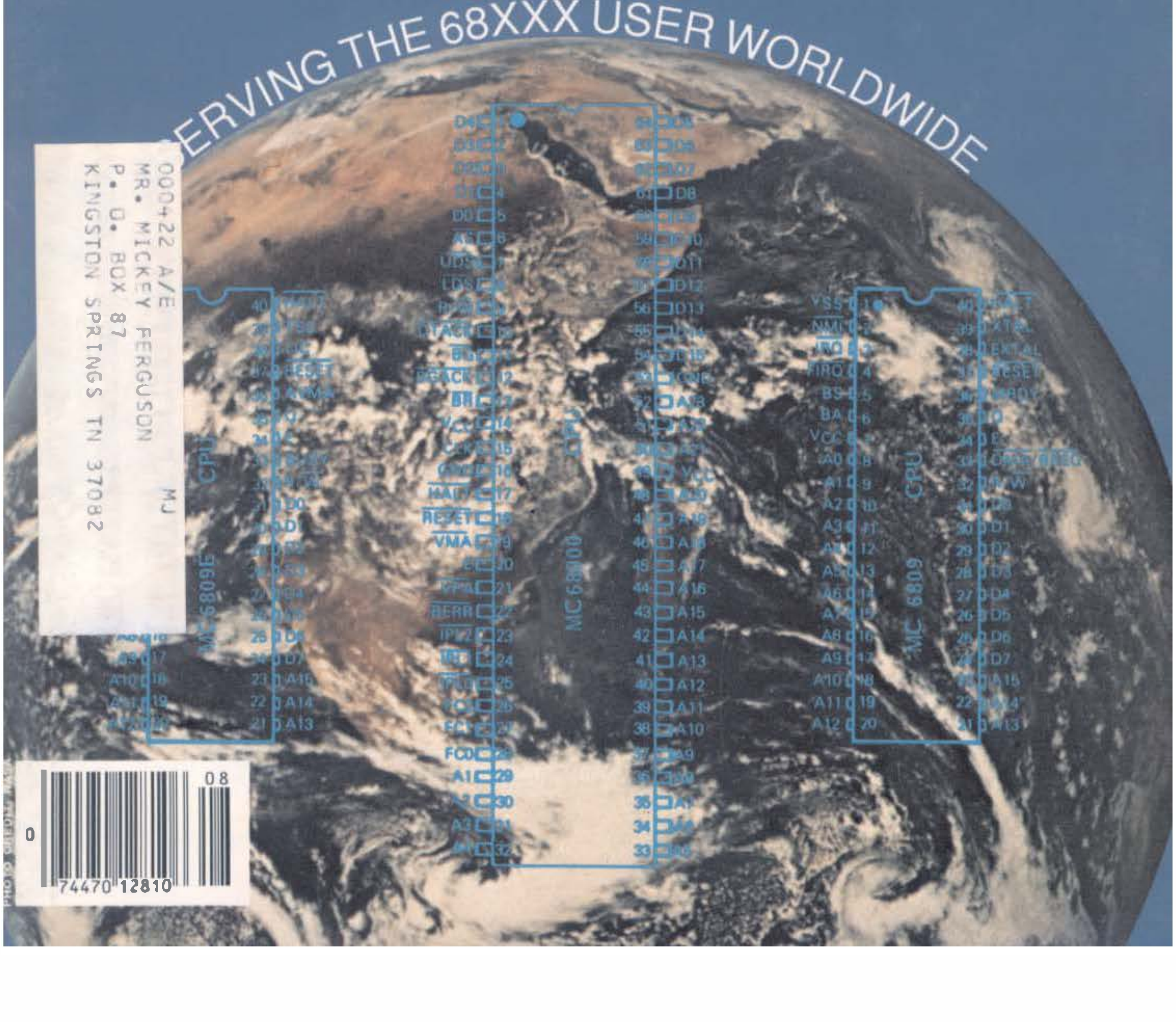
The Magazine for Motorola CPU Devices *For Over a Decade!*

This Issue:

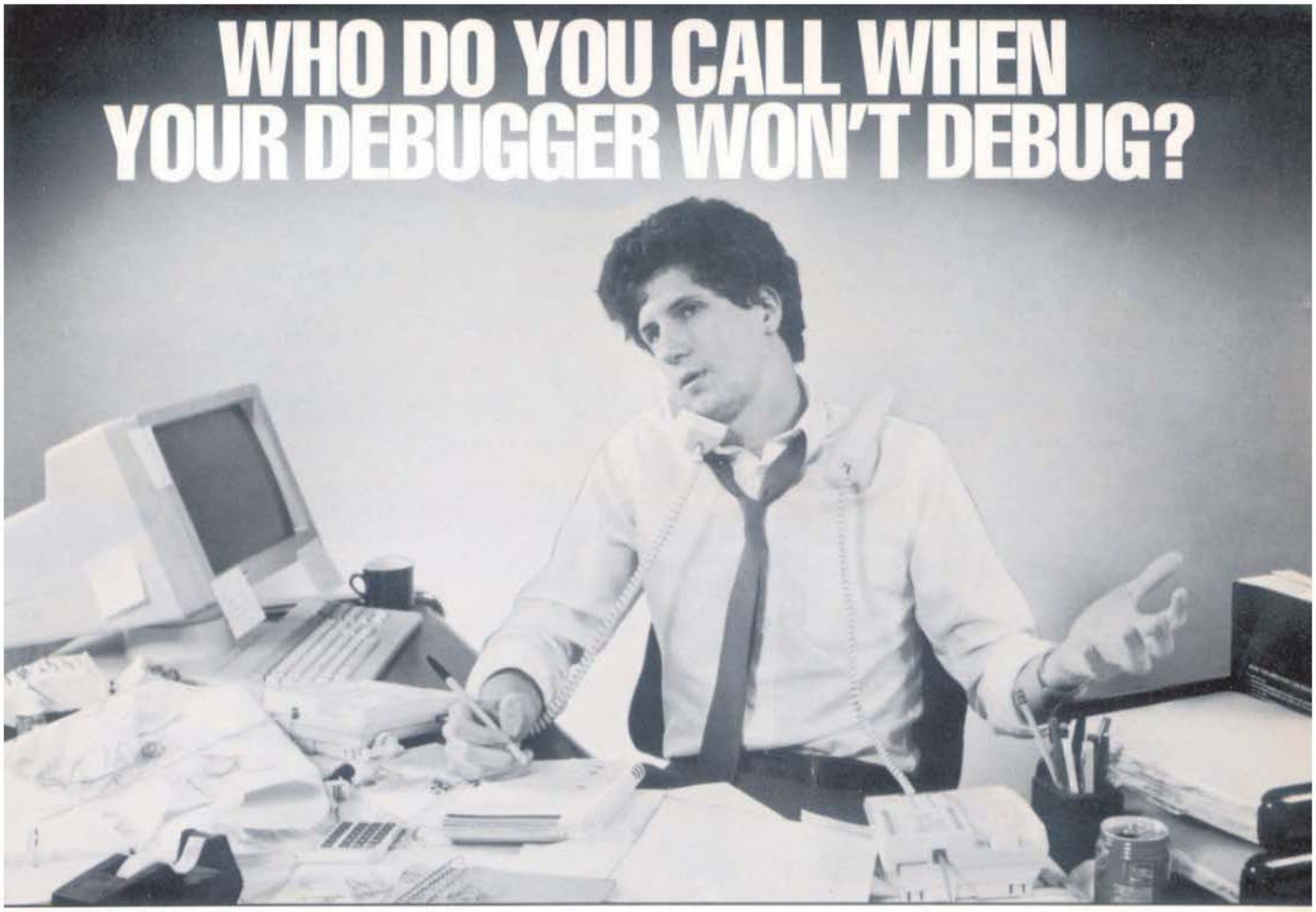
OS-9 SK*DOS Atari Amiga FLEX Macintosh *A User Contributor Journal* **And Lots More!**

VOLUME X ISSUE VIII ● Devoted to the 68XXX User ● August 1988

The Grandfather of "DeskTop Publishing™"



000+22 A/E MJ
MR. MICKEY FERGUSON
P. O. BOX 87
KINGSTON SPRINGS TN 37082



The problem with most real-time operating systems is simple, they're not an integrated solution. You end up dealing with a multitude of suppliers for languages, compilers, debuggers and other important development tools. And when something does go wrong, it can be a frustrating experience trying to straighten out the mess.

### *Why Not Try the Microware One-Stop Total Solution?*

Microware's OS-9 Real-Time Operating System is a total integrated software system, not just a kernel. We offer an extensive set of development tools, languages, I/O and Kernel options. ***And this total integrated solution is entirely designed, built and supported by the same expert Microware team.***

Microware is a registered trademark of Microware Systems Corporation.
OS-9 is a trademark of Microware.
UNIX is a trademark of AT&T.
VAX is a trademark of DEC.

### The OS-9 Success Kit

*A Total Integrated Solution for Your Next Project*

**Development Tools:**
C Source Level Debugger
Symbolic Debugger
System State Debugger
uMACS Text Editor
Electronic Mail
Communications
Super Shell

**Kernel Options:**
MMU (Security Protection) Support
Math Coprocessor Support

**Languages:**
C*
Basic
Pascal
Fortran
Ada**
Assembler*

**I/O Options:**
SCSI, SASI & SMD Disks
3-, 5-, 8-inch Diskettes
Magnetic Tape
Ethernet - TCP/IP
Arcnet - OS-9/Net

*Resident or UNIX versions available
**VAX hosted

### *Modularity Lets YOU Choose Just What You Need.*

The modular design of OS-9 allows our Operating System to adapt as your requirements change. OS-9 can support a complete spectrum of applications — from embedded ROM-based code in board-level products all the way up to large-scale systems.

### *Support is Part of the Package.*

Microware is proudly setting the industry's standard for customer support. You'll find professional and comprehensive technical documentation and a Customer Hotline staffed by courteous and authoritative software engineers.

So stop messing with simple kernels and independent suppliers. Call Microware today and find out more about the "One-Stop Integrated Solution" with OS-9!

microware® OS-9

**Microware Systems Corporation**
1900 N.W. 114th Street
Des Moines, Iowa 50322
Phone: 515/224-1929

**Western Regional Office**
4401 Great America Parkway
Santa Clara, California 95054
Phone: 408/980-0201

**Microware Japan Ltd.**
41-19 Honcho 4-Chome
Funabashi City
Chiba 273, Japan
Phone: 0474 (22) 1747

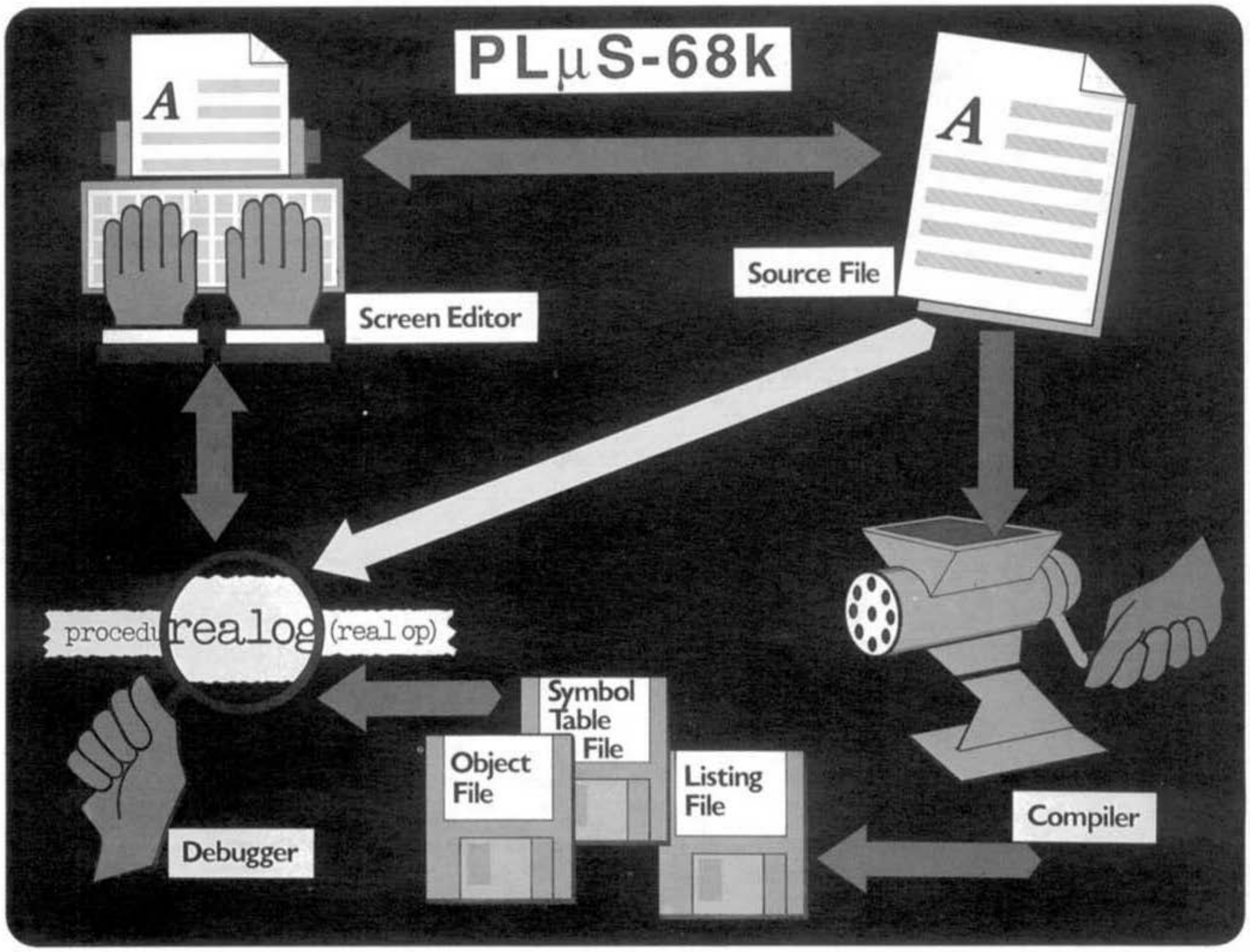
# PLμS-68k

**A Complete Program Development Environment**

**MC68000 - MC68010 - MC68020/881 Processors**

**OS-9 User-state or System-state Modules**

**Stand Alone ROM Based Targets**



**WINDRUSH**
**Micro Systems Ltd.**
Worstead Laboratories, North Walsham,
Norfolk NR28 9SA, England
TEL: (44) (692) 404086
FAX: (44) (692) 404091
TLX: 975548 WMICRO G

**PLμS-68k**
for 68008/68000/68010 Targets **$375.00**
**PLμS-020**
for 68008/68000/68010
& 68020/881 Targets **$475.00**

*Prices include Air Mail Postage, Insurance and one year FREE updates.*

**A Member of the CPI Family**

# 68 Micro Journal

*10 Years of Dedication to Motorola CPU Users*

**6800 6809 68000 68010 68020**

**The Originator of "DeskTop Publishing™"**

**Publisher**
Don Williams Sr.

**Executive Editor**
Larry Williams

**Production Manager**
Tom Williams

**Office Manager**
Joyce Williams

**Subscriptions**
Stacy Power

***Contributing & Associate Editors***

| | |
|---|---|
| Ron Anderson | Dr. E.M. "Bud" Pass |
| Ron Voigts | Art Weller |
| Doug Lurie | Dr. Theo Elbert |
| Ed Law | & Hundreds More of Us |

## Contents

**68 MICRO JOURNAL**

*"Contribute Nothing - Expect Nothing"* DMW 1986

***COMPUTER PUBLISHING, INC.***

*"Over a Decade of Service"*

"World Wide"

**68 MICRO JOURNAL**
**Computer Publishing Center**
**5900 Cassandra Smith Road**
**PO Box 849**
**Hixson, TN 37343**
**Phone (615) 842-4600 Telex 510 600-6630**

Copyrighted © 1987 by Computer Publish ng, Inc.

68 Micro Journal s the *original* "DeskTop Publishing" product and has continously published since 1978 using only micro-computers and special "DeskTop" software. Using first a it built 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 M cro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

68 Micro Journal is published 12 t mes a year by Computer Publishing Inc. Second Class Postage paid ISSN 0194-5025 at Hixson, TN. and additional entries. Postmaster: send form 3597 to 68 Micro Journal, POB 849, Hixson, TN 37343.

**Subscription Rates**

1 Year $24.50 USA, Canada & Mexico $34.00 a year. Others add $12.00 a year surface, $48.00 a year Airmail, USA funds. 2 years $42.50, 3 years $64.50 plus additional postage for each additional year.

**Items or Articles for Publication**

Articles subm tted for publ cation must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles submitted should be on diskette, OS-9, SK*DOS, FLEX, Macintosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

*Please - do not format with spaces any text indents, charts, etc. (source listing o.k.). We will edit in all formatting. Text should fall flush left and use a carriage return only to indicate a paragraph end. Please write for free authors guide.*

**Letters & Advertising Copy**

Letters to the Editor should be the original copy, signed! Letters of grip as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of $15.5 for first 15 words. Add $.60 per word thereafter. No classifieds accepted by telephone.

**PAT - JUST**

**PAT**

**With 'C' Source**

**$229.00**

All OS-9
68XXX
Systems

service For Over a Decade!
1,200 Titles +

A Division of

CPI

**S.E. Media**

3900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

6800-6809-68008-68000-68010-68020

**PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR** with all the best of PIE. For those who swore by and loved **PIE**, this is for **YOU! All PIE features & much more!** Too many features to list. And if you don't like ours, change or add your own. **C source included.** Easily configured to your CRT terminal, with special configuration section. No sweat!

**68008 - 68000 - 68010 - 68020 OS-9 68K $229.00**

**COMBO**

**PAT/JUST**

**Special $249.00**

**JUST**

**JUST from S. E. MEDIA - -** Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

**68008 - 68000 - 68010 - 68020 OS-9 68K**
**With 'C' source $79.95**

# MUSTANG -020 Super SBC ™

## DATA-COMP Proudly Presents the First Under $4300 "SUPER MICRO" See other advertising (backcover) for economy system (68008) - under $2400 complete!

The MUSTANG-020 68020 SBC provides a powerful, compact, 32 bit computer system featuring the "state of the art" Motorola 68020 "super" micro-processor. It comes standard with 2 megabyte of high-speed SIP dynamic RAM, serial and parallel ports, floppy disk controller, a SASI hard disk interface for intelligent hard disk controllers and a battery backed-up time-of-day clock. Provisions are made for the super powerful Motorola MC68881 floating point math co-processor, for heavy math and number crunching applications. An optional network interface uses one serial (four (4) standard, expandable to 20) as a 125/bit per second network channel. Supports as many as 32 nodes.

The MUSTANG-020 is ideally suited to a wide variety of applications. It provides a cost effective alternative to the other MC68020 systems now available. It is an excellent introductory tool to the world of hi-power, hi-speed new generation "super micros". In practical applications it has numerous applications, ranging from scientific to education. It is already being used by government agencies, labs, universities, business and practically every other critical applications center, worldwide, where true multi-user, multi-tasking needs exist. The MUSTANG-020 is UNIX C level V compatible. Where low cost and power is a must, the MUSTANG-020 is the answer, as many have discovered. Proving that price is not the standard for quality!

As a software development station, a general purpose scientific or small to medium business computer, or a super efficient real-time controller in process control, the MUSTANG-020 is the cost effective choice. With the optional MC68881 floating point math co-processor installed, it has the capability of systems costing many times over it's total acquisition cost.

With the DATA-COMP "total package", consisting of a heavy duty metal cabinet, switching power supply with rf/line by-passing, 5 inch DS/DD 80 track floppy, Xebec hard disk controller, 25 megabyte winchester hard disk, four serial RS-232 ports and a UNIX C level V compatible multi-tasking, multi-user operating system, the price is under $4300, w/12.5 megahertz system clock (limited time offer). Most all popular high level languages are available at very reasonable cost. The system is expandable to 32serial ports, at a cost of less than $65 per port, in multiples of 8 port expansion options.

The SBC fully populated, quality tested, with 4 serial ports pre-wired and board mounted is available for less that $2500. Quantity discounts are available for OEM and special applications, in quantity. All that is required to bring to complete "system" standards is a cabinet, power supply, disks and operating system. All these are available as separate items from DATA-COMP.

*Available 12.5-25 Mhz systems, call for special prices*

A special version of the Motorola 020-BUG is installed on each board. 020-BUG is a ROM based debugger package with facilities for downloading and executing user programs from a host system. It includes commands for display and modification of memory, breakpoint capabilities, a powerful assembler/disassemble and numerous system diagnostics. Various 020-BUG system routines, such as I/O handlers are available for user programs.

Normal system speed is 3-4.5 MIPS, with burst up to 10 MIPS, at 16.6 megahertz. Intelligent I/O available for some operating systems.

Hands-on "actual experience sessions", before you buy, are available from DATA-COMP. Call or write for additional information or pricing.

**Data-Comp Division**

A Decade of Quality Service"

Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

## Mustang-020 Mustang-08 Benchmarks

| | 32 bit Integer | Register Long |
|---|---|---|
| IBM AT 7300 Xenix Sys 3 | 9.7 | |
| AT&T 7300 UNIX PC 68010 | 7.2 | 4.3 |
| DEC VAX 11/780 UNIX Berkley 4.2 | 3.6 | 3.2 |
| DEC VAX 11/750 " " | 5.1 | 3.2 |
| 68008 OS-9 68K 8 Mhz | 18.0 | 9.0 |
| 68000 OS-9 68K 10 Mhz | 6.5 | 4.0 |
| *MUSTANG-08 68008 OS-9 68K 10 Mhz* | *9.8* | *6.5* |
| *MUSTANG-020 68020 OS-9 68K 16 Mhz* | *2.2* | *0.88* |
| *MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz* | *1.8* | *1.22* |

```
Main()
            {
                register long i;
                for (i=0; i < 999999; ++i);
            }
```

Estimated MIPS - MUSTANG-020 — 4.5 MIPS,
Burst to 8 - 10 MIPS: Motorola Specs

## OS-9

| | |
|---|---|
| OS-9 Professional Ver | $850.00 |
| *Includes C Compiler | |
| Basic09 | 300.00 |
| C Compiler | 300.00 |
| 68000 Disassembler (w/source add: $100.00) | 100.00 |
| Fortran 77 | 750.00 |
| Microware Pascal | 500.00 |
| Omegasoft Pascal | 900.00 |
| Stylo-Graph | 495.00 |
| Stylo-Spell | 195.00 |
| Stylo-Merge | 175.00 |
| Stylo-Graph-Spell-Merge | 695.00 |
| PAT w/C source | 229.00 |
| JUST w/C source | 79.95 |
| PAT/JUST Combo | 249.50 |
| Sculptor+ (see below) | 995.00 |
| COM | 125.00 |

## UniFLEX

| | |
|---|---|
| UniFLEX (68020 ver) | $450.00 |
| Screen Editor | 150.00 |
| Sort-Merge | 200.00 |
| BASIC/PreCompiler | 300.00 |
| C Compiler | 350.00 |
| COBOL | 750.00 |
| CMODEM w/source | 100.00 |
| TMODEM w/source | 100.00 |
| X-TALK (see Ad) | 99.95 |
| Cross Assembler | 50.00 |
| Fortran 77 | 450.00 |
| Sculptor+ (see below) | 995.00 |

Standard MUSTANG-020™ shipped 12.5 Mhz.

| | |
|---|---|
| Add for 16.6 Mhz 68020 | 375.00 |
| Add for 16.6 Mhz 68881 | 375.00 |
| Add for 20 Mhz 68020/RAM | 750.00 |
| 16 Port exp. RS-232 | 335.00 |
| Requires 1 or 2 Adapter Cards below RS232 Adapter | 165.00 |

Each card supports 4 additional ser. ports
(total of 36 serial ports supported)

| | |
|---|---|
| 60 line Parallel I/O card | 398.00 |

Uses 3 68230 Interface/Timer chips,
6 groups of 8 lines each, separate buffer
direction control for each group.

| | |
|---|---|
| Prototype Board | 75.00 |

areas for both dip and PGA devices & a
pre-wired memory area up to 512K DRAM.

| | |
|---|---|
| SBC-AN | 475.00 |

Interface between the system and
ARCNET modified token-passing LAN. fiber optics optional - call.

| | |
|---|---|
| LAN software drivers | 120.00 |
| Expansion for Motorola I/O Channel Modules | $195.00 |

Special for complete MUSTANG-020™ system buyers - Sculptor+
$495.00 SAVE $500.00
Software Discounts

All MUSTANG-020™ system and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quotes. Discounts apply after the sale as well.

# The PRO!

## Mustang Specifications

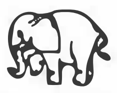
12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
32-bit wide data and address buses, non-multiplexed
on chip instruction cache
object code compatible with all 68XXX family processors
enhanced instruction set - math co-processor interface
68881 math hi-speed floating point co-processor (optional)
direct extension of full 68020 instruction set
full support IEEE P754, draft 10.0
transcendental and other scientific math functions
2 Megabyte of SIP RAM (512 x 32 bit organization)
up to 256K bytes of EPROM (64 x 32 bits)
4 Asynchronous serial I/O ports standard
optional to 20 serial ports
standard RS-232 interface
optional network interface
Timed 8 bit parallel port (1/2 MC68230)
Centronics type pinout
expansion connector for I/O devices
16 bit data path
256 byte address space
2 interrupt inputs
clock and control signals
Motorola I/O Channel Modules
time of day clock/calendar w/battery backup
controller for 2, 5 1/4" floppy disk drives
single or double side, single or double density
35 to 80 track selectable (48-96 TPI)
SASI interface
programmable periodic interrupt generator
interrupt rate from micro-seconds to seconds
highly accurate time base (5 PPM)
5 bit sense switch, readable by the CPU
Hardware single-step capability

TM CPI
*Mustang Hi-Speed Systems*
*Only from Data-Comp Div.*
68008-68030

***Don't be mislead!***
***ONLY Data-Comp delivers the Super MUSTANG-020***

These hi-speed 68020 systems are presently working at NASA, Atomic Energy Commission, Government Agencies as well as Universities, Business, Labs, and other Critical Application Centers, worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level V compatability and low cost is a must.

Only the "PRO" Version of OS-9 Supported!



This is *HEAVY DUTY* Country!

For a limited time we will offer a $400 trade-in on your old 68XXX SBC. Must be working properly and complete with all software, cables and documentation.
Call for more information

**Price List:**

| | | |
|---|---|---|
| Mustang-020 SBC | | $2490.00 |
| Cabinet w/switching PS | | $299.95 |
| 5"-80 track floppy DS/DD | | $269.95 |
| Floppy Cable | | $39.95 |
| OS-9 68K Professional Version | | $850.00 |
| C Compiler ($500 Value) | | N/C |
| Winchester Cable | | $39.95 |
| Winchester Drive 25 Mbyte | | $895.00 |
| Hard Disk Controller | | $395.00 |
| Shipping USA UPS | | $20.00 |
| UniFLEX | Less | $100.00 |
| MC68881 f/p math processor | Add | $275.00 |
| 16.67 Mhz MC68020 | | $375.00 |
| 16.67 Mhz MC68881 | | $375.00 |
| 20 Mhz MC68020 Sys | | $750.00 |
| Note all 68881 chips work with 20 Mhz Sys | | |
| | Total: | $5299.80 |

**Save $1000.00**
**Complete**
**25 Mbyte HD System**
**$4299.80**
**85Mbyte HD System**
**$5748.80**

Note: Only Professional OS-9 Now Available (68020 Version)
Includes ($500) C Compiler - 68020 & 68881 Supported -
For UPGRADES Write or Call for Professional OS-9 Upgrade Kit

**Data-Comp Division**
CPI A Decade of Quality Service"
*Systems World-Wide*
**Computer Publishing, Inc.** 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

C

The C Programmers Reference Source. Always Right On Target!

# C User Notes

A Tutorial Series

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
*Computer Systems Consultants*

**INTRODUCTION**

This chapter continues the discussion of dbug, a C debugging package. It is a useful tool for debugging and testing C programs. It was developed by Fred Fish, who placed it into the public domain. The C code for the dbug package and for extensions to it appear in this and subsequent chapters.

DBUG.H

The Dbug.h header file must be included in any program in case Fred Fish's dbug package is to be used. It contains the appropriate preprocessor commands to call support routines in the dbug run-time library contained in dbug.c.

To disable compilation of the dbug commands, define the preprocessor symbol "DBUG_OFF". This will result in null expansions so that the resulting code will be smaller.

All externally visible symbol names follow the pattern _db_xxx..xx_ to minimize the possibility of a dbug package symbol coinciding with a user-defined symbol.

```
/*
 *      dbug.h    header file for dbug package
 *
 *      Fred Fish
 */

/*
 *      Internally used variables which must be global.
 */

#ifndef DBUG_OFF
extern int _db_on_;           /* TRUE if debug currently enabled */
extern FILE *_db_fp_;         /* Current debug output stream */
extern char *_db_process_;    /* Name of current process */
extern int _db_keyword_();    /* Accept/reject keyword */
extern void _db_push_();      /* Push state, set up new state */
extern void _db_pop_();       /* Pop previous debug state */
extern void _db_enter_();     /* New user function entered */
extern void _db_return_();    /* User function return */
extern void _db_pargs_();     /* Remember args for line */
extern void _db_doprnt_();    /* Print debug output */
extern void _db_setjmp_();    /* Save debugger environment */
extern void _db_longjmp_();   /* Restore debugger environment */
#endif
```

```
/*
 *      These macros provide a user interface into functions in the
 *      dbug runtime support library.  They isolate users from changes
 *      in the MACROS and/or runtime support.
 *
 *      The symbols "__LINE__" and "__FILE__" are expanded by the
 *      preprocessor to the current source file line number and file
 *      name respectively.
 *
 *      WARNING -  Because the DBUG_ENTER macro allocates space on
 *      the user function's stack, it must follow any declarations and
 *      precede any executable statements in the user  function.
 *
 */

#ifdef DBUG_OFF
#define DBUG_ENTER(a1)
#define DBUG_EXECUTE(keyword,a1)
#define DBUG_FILE(stderr)
#define DBUG_LONGJMP longjmp
#define DBUG_POP()
#define DBUG_PRINT(keyword,arglist)
#define DBUG_PROCESS(a1)
#define DBUG_PUSH(a1)
#define DBUG_RETURN(a1) return(a1)
#define DBUG_SETJMP setjmp
#define DBUG_VOID_RETURN return
#else
#define DBUG_ENTER(a) char *_db_func_, *_db_file_; int _db_level_; \
      _db_enter_(a,__FILE__,__LINE__,&_db_func_,&_db_file_,&_db_level_)
#define DBUG_EXECUTE(keyword,a1) \
     { if (_db_on_) { if (_db_keyword_(keyword)) { a1 } } }
#define DBUG_FILE (_db_fp_)
#define DBUG_LEAVE (_db_return_(__LINE__, &_db_func_, &_db_file_,
&_db_level_))
#define DBUG_LONGJMP(a1,a2) (_db_longjmp_(), longjmp(a1, a2))
#define DBUG_POP() _db_pop_()
#define DBUG_PRINT(keyword,arglist) \
     { if (_db_on_) { _db_pargs_(__LINE__,keyword); _db_doprnt_ ar-
glist; } }
#define DBUG_PROCESS(a1) (_db_process_ = a1)
#define DBUG_PUSH(a1) _db_push_(a1)
#define DBUG_RETURN(a1) {DBUG_LEAVE; return(a1);}
#define DBUG_SETJMP(a1) (_db_setjmp_(), setjmp(a1))
#define DBUG_VOID_RETURN {DBUG_LEAVE; return;}
#endif
```

DBUG.C

The code below represents the runtime library supporting the dbug package.

```
/*
 *      dbug.c   runtime support routines for dbug package
 *
 *      These are the runtime support routines for the dbug package.
 *      The dbug package has two main components; the user include
 *      file containing various macro definitions, and the runtime
 *      support routines which are called from the macro expansions.
 *
 *      Externally visible functions in the runtime support module
 *      use the naming convention pattern "_db_xx...xx_", thus
 *      they are unlikely to collide with user defined function names.
 *
 *      Fred Fish               (base code)
 *      (Currently at Motorola Computer Division, Tempe, Az.)
 *      hao!noao!mcdsun!fnf
 *      (602) 438-3614
 *
 *      Binayak Banerjee        (profiling enhancements)
 *      seismo!bpa!sjuvax!bbanerje
 */

#include <stdio.h>

#ifdef AMIGA
#define HZ (50)                 /* Probably in some header somewhere
*/
#endif

/*
 *      Manifest constants that should not require any changes.
 */

#define FALSE           0       /* Boolean FALSE */
#define TRUE            1       /* Boolean TRUE */
#define EOS             0       /* End Of String marker */

/*
 *      Manifest constants which may be "tuned" if desired.
 */

#define PRINTBUF        1024    /* Print buffer size */
#define INDENT          4       /* Indentation per trace level */
#define MAXDEPTH        200     /* Maximum trace depth default */

/*
 *      The following flags are used to determine which
 *      capabilities the user has enabled with the state push macro.
 */
```

```
    #define DEBUG_ON        000002  /* Debug enabled */
    #define DEPTH_ON        000020  /* Function nest level print enabled
*/
    #define FILE_ON         000004  /* File name print enabled */
    #define LINE_ON         000010  /* Line number print enabled */
    #define NUMBER_ON       000100  /* Number each line of output */
    #define PROCESS_ON      000040  /* Process name print enabled */
    #define PROFILE_ON      000200  /* Print out profiling mode */
    #define TRACE_ON        000001  /* Trace enabled */

    #define DEBUGGING (stack -> flags & DEBUG_ON)
    #define PROFILING (stack -> flags & PROFILE_ON)
    #define STREQ(a,b) (strcmp(a,b) == 0)
    #define TRACING (stack -> flags & TRACE_ON)

    /*
     *      Make it easy to change storage classes if necessary.
     */

    #define AUTO auto               /* Names to be allocated on stack */
    #define BOOLEAN int             /* True or false */
    #define EXPORT                  /* Allocated here, available globally
*/
    #define IMPORT extern           /* Names defined externally */
    #define LOCAL static            /* Names not needed by outside world
*/
    #define REGISTER register       /* Names to be placed in registers */
    #define VOID void               /* Some classes have no class */

    /*
     *      The following define is for the variable arguments kluge.
     *      See the comments in _db_doprnt_().
     *
     *      Also note that the longer this list, the less prone to failing
     *      on long argument lists, but the more stuff that must be moved
     *      around for each call to the runtime support routines.  The
     *      length may really be critical if the machine convention is
     *      to pass arguments in registers.
     *
     *      Note that the default define allows up to 16 integral argu-
ments,
     *      or 8 floating point arguments (doubles), on most machines.
     *
     */

    #define ARGLIST a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15

    /*
     * The default file for profiling.
     */

    #define PROF_FILE       "dbugmon.out"
```

```
    /*
     *      Variables which are available externally but should only
     *      be accessed via the macro package facilities.
     */

    EXPORT BOOLEAN _db_on_ = FALSE;         /* TRUE if debugging currently
on */
    EXPORT BOOLEAN _db_pon_ = FALSE;        /* TRUE if debugging currently
on */
    EXPORT FILE *_db_fp_ = stderr;          /* Output stream, default
stderr */
    EXPORT FILE *_db_pfp_ = (FILE *)0;      /* Profile stream,
'dbugmon.out' */
    EXPORT char *_db_process_ = "dbug";     /* Pointer to process name;
argv[0] */

    /*
     *      Externally supplied functions.
     */

    #ifdef unix                     /* Only needed for unix */
    IMPORT VOID perror ();          /* Print system/library error */
    IMPORT int access ();           /* Test file for access */
    IMPORT int chown ();            /* Change owner of a file */
    IMPORT int getgid ();           /* Get real group id */
    IMPORT int getuid ();           /* Get real user id */
    #else
    #if !(AMIGA && LATTICE)
    LOCAL VOID perror ();           /* Fake system/library error print
routine */
    #endif
    #endif

    #if BSD4_3 || sun
    IMPORT int getrusage ();
    #endif

    IMPORT FILE *fopen ();          /* Open a stream */
    IMPORT VOID exit ();            /* Terminate execution */
    IMPORT VOID free ();
    IMPORT char *malloc ();         /* Allocate memory */
    IMPORT char *strcpy ();         /* Copy strings around */
    IMPORT int atoi ();             /* Convert ascii to integer */
    IMPORT int fclose ();           /* Close a stream */
    IMPORT int fprintf ();          /* Formatted print on file */
    IMPORT int strcmp ();           /* Compare strings */
    IMPORT int strlen ();           /* Find length of string */

    #ifndef fflush                  /* This is sometimes a macro */
    IMPORT int fflush ();           /* Flush output for stream */
    #endif
    /*
     *      The user may specify a list of functions to trace or
     *      debug.  These lists are kept in a linear linked list,
     *      a very simple implementation.
     */

    struct link
    {
        char *string;               /* Pointer to link's contents */
        struct link *next_link;     /* Pointer to the next link */
    };

    /*
     *      Debugging states can be pushed or popped off of a
     *      stack which is implemented as a linked list.  Note
     *      that the head of the list is the current state and the
     *      stack is pushed by adding a new state to the head of the
     *      list or popped by removing the first link.
     */

    struct state
    {
        FILE *out_file;                     /* Current output stream */
        FILE *prof_file;                    /* Current profiling stream */
        int flags;                          /* Current state flags */
        int level;                          /* Current function nesting
level */
        int maxdepth;                       /* Current maximum trace depth
*/
        struct link *functions;             /* List of functions */
        struct link *keywords;              /* List of debug keywords */
        struct link *p_functions;           /* List of profiled functions
*/
        struct link *processes;             /* List of process names */
        struct state *next_state;           /* Next state in the list */
        unsigned int delay;                 /* Delay after each output
line */
    };

    LOCAL struct state *stack = NULL;       /* Linked list of stacked
states */

    /*
     *      Local variables not seen by user.
     */

    LOCAL BOOLEAN init_done = FALSE;/* Set to TRUE when initialization
done */
    LOCAL char *file = "?file";     /* Name of current user file */
    LOCAL char *func = "?func";     /* Name of current user function */
    LOCAL int lineno = 0;           /* Current debugger output line number
*/

    #if unix || AMIGA
    LOCAL char *jmpfile;            /* Remember current file for setjmp */
    LOCAL char *jmpfunc;            /* Remember current function for
setjmp */
    LOCAL int jmplevel;             /* Remember nesting level at setjmp ()
*/
    #endif

    LOCAL BOOLEAN DoTrace ();       /* Test for tracing enabled */
    LOCAL BOOLEAN Writable ();      /* Test to see if file is writable */
    LOCAL VOID ChangeOwner ();      /* Change file owner and group */
    LOCAL VOID CloseFile ();        /* Close debug output stream */
    LOCAL VOID DoPrefix ();         /* Print debugger line prefix */
    LOCAL VOID FreeList ();         /* Free memory from linked list */
    LOCAL VOID Indent ();           /* Indent line to specified indent */
    LOCAL VOID OpenFile ();         /* Open debug output stream */
    LOCAL VOID OpenProfile ();      /* Open profile output stream */
    LOCAL VOID PushState ();        /* Push current debug state */
    LOCAL char *BaseName ();        /* Remove leading pathname components
*/
    LOCAL char *DbugMalloc ();      /* Allocate memory for runtime support
*/
    LOCAL char *StrDup ();          /* Make a fresh copy of a string */
    LOCAL struct link *ListParse ();/* Parse a debug command string */
    LOCAL unsigned long Clock ();   /* Return current user time (ms) */

    /* Supplied in Sys V runtime environ */
    LOCAL char *strrchr ();         /* Find last occurance of char */
    LOCAL char *strtok ();          /* Break string into tokens */

    /*
     *      The following local variables are used to hold the state
information
     *      between the call to _db_pargs_() and _db_doprnt_(), during
     *      expansion of the DBUG_PRINT macro.  This is the only macro
     *      that currently uses these variables.  The DBUG_PRINT macro
     *      and the new _db_doprnt_() routine replace the older DBUG_N
macros
     *      and their corresponding runtime support routine _db_printf_().
     *
     *      These variables are currently used only by _db_pargs_() and
     *      _db_doprnt_().
     */

    LOCAL char *u_keyword = "?";    /* Keyword for current macro */
    LOCAL int u_line = 0;           /* User source code line number */

    /*
     *      Miscellaneous printf format strings.
     */

    #define ERR_ABORT "%s: debugger aborting because %s\n"
    #define ERR_CHOWN "%s: can't change owner/group of \"%s\": "
    #define ERR_CLOSE "%s: can't close debug file: "
    #define ERR_MISSING_RETURN "%s: missing DBUG_RETURN macro in function
```

```
\"%s\"\n"
    #define ERR_OPEN "%s: can't open debug output stream \"%s\": "
    #define ERR_PRINTF "%s: obsolete object file for '%s', please
recompile!\n"

    /*
     *      Macros and defines for testing file accessibility under UNIX.
     */

    #ifdef unix
    #define A_EXECUTE     01              /* Test for execute permission
*/
    #define A_EXISTS      00              /* Test for file existance */
    #define A_READ        03              /* Test for read access */
    #define A_WRITE       02              /* Test for write access */
    #define EXISTS(pathname) (access (pathname, A_EXISTS) == 0)
    #define WRITABLE(pathname) (access (pathname, A_WRITE) == 0)
    #else
    #define EXISTS(pathname) (FALSE)      /* Assume no existance */
    #endif

    /*
     *      Translate some calls among different systems.
     */

    #ifdef unix
    #define Delay sleep
    IMPORT unsigned int sleep ();   /* Pause for given number of seconds
*/
    #endif

    #ifdef AMIGA
    IMPORT int Delay ();            /* Pause for given number of ticks */
    #endif


    /*
     *      _db_push_     push current debugger state and set up new one
     *
     *      VOID _db_push_ (control)
     *      char *control;
     *
     *      Given pointer to a debug control string in "control", pushes
     *      the current debug state, parses the control string, and sets
     *      up a new debug state.
     *
     *      The only attribute of the new state inherited from the previ-
ous
     *      state is the current function nesting level.  This can be
     *      overridden by using the "r" flag in the control string.
     *
     *      The debug control string is a sequence of colon separated
fields
     *      as follows:
```

```
     *
     *              <field_1>:<field_2>:...:<field_N>
     *
     *      Each field consists of a mandatory flag character followed by
     *      an optional "," and comma separated list of modifiers:
     *
     *              flag[,modifier,modifier,...,modifier]
     *
     *      The currently recognized flag characters are:
     *
     *              d       Enable output from DBUG_<N> macros for
     *                      for the current state.  May be followed
     *                      by a list of keywords which selects output
     *                      only for the DBUG macros with that keyword.
     *                      A null list of keywords implies output for
     *                      all macros.
     *
     *              D       Delay after each debugger output line.
     *                      The argument is the number of tenths of
seconds
     *                      to delay, subject to machine capabilities.
     *                      I.E.  -#D,20 is delay two seconds.
     *
     *              f       Limit debugging and/or tracing, and profiling
to the
     *                      list of named functions.  Note that a null
list will
     *                      disable all functions.  The appropriate "d" or
"t"
     *                      flags must still be given, this flag only
limits their
     *                      actions if they are enabled.
     *
     *              F       Identify the source file name for each
     *                      line of debug or trace output.
     *
     *              g       Enable profiling.  Create a file called
'dbugmon.out'
     *                      containing information that can be used to
profile
     *                      the program.  May be followed by a list of
keywords
     *                      that select profiling only for the functions
in that
     *                      list.  A null list implies that all functions
are
     *                      considered.
     *
     *              L       Identify the source file line number for
     *                      each line of debug or trace output.
     *
     *              n       Print the current function nesting depth for
     *                      each line of debug or trace output.
     *
```

```
 *           N      Number each line of dbug output.
 *
 *           p      Limit debugger actions to specified processes.
 *                  A process must be identified with the
 *                  DBUG_PROCESS macro and match one in the list
 *                  for debugger actions to occur.
 *
 *           P      Print the current process name for each
 *                  line of debug or trace output.
 *
 *           r      When pushing a new state, do not inherit
 *                  the previous state's function nesting level.
 *                  Useful when the output is to start at the
 *                  left margin.
 *
 *           t      Enable function call/exit trace lines.
 *                  May be followed by a list (containing only
 *                  one modifier) giving a numeric maximum
 *                  trace level, beyond which no output will
 *                  occur for either debugging or tracing
 *                  macros.  The default is a compile time
 *                  option.
 *
 *      Some examples of debug control strings which might appear
 *      on a shell command line (the "-#" is typically used to
 *      introduce a control string to an application program) are:
 *
 *           -#d:t
 *           -#d:f,main,subr1:F:L:t,20
 *           -#d,input,output,files:n
 *
 *      For convenience, any leading "-#" is stripped off.
 */


VOID _db_push_ (control)
char *control;
{
    REGISTER char * scan;
    REGISTER struct link * temp;

    if (control && *control == '-')
    {
        if (*++control == '#')
            control++;
    }
    control = StrDup (control);
    PushState ();
    scan = strtok (control, ":");
    for (; scan != NULL; scan = strtok ((char *)NULL, ":"))
    {
        switch (*scan++)
        {
        case 'd':
            _db_on_ = TRUE;
            stack ->flags |= DEBUG_ON;
            if (*scan++ == ',')
                stack ->keywords = ListParse (scan);
            break;
        case 'D':
            stack ->delay = 0;
            if (*scan++ == ',')
            {
                temp = ListParse (scan);
                stack ->delay = DelayArg (atoi (temp ->string));
                FreeList (temp);
            }
            break;
        case 'f':
            if (*scan++ == ',')
                stack ->functions = ListParse (scan);
            break;
        case 'F':
            stack ->flags |= FILE_ON;
            break;
        case 'g':
            _db_pon_ = TRUE;
            OpenProfile(PROF_FILE);
            stack ->flags |= PROFILE_ON;
            if (*scan++ == ',')
                stack ->p_functions = ListParse (scan);
            break;
        case 'L':
            stack ->flags |= LINE_ON;
            break;
        case 'n':
            stack ->flags |= DEPTH_ON;
            break;
        case 'N':
            stack ->flags |= NUMBER_ON;
            break;
        case 'o':
            if (*scan++ == ',')
            {
                temp = ListParse (scan);
                OpenFile (temp ->string);
                FreeList (temp);
            }
            else {
                OpenFile ("-");
            }
            break;
        case 'p':
            if (*scan++ == ',')
                stack ->processes = ListParse (scan);
            break;
        case 'P':
            stack ->flags |= PROCESS_ON;
            break;
        case 'r':
            stack ->level = 0;
            break;
        case 't':
            stack ->flags |= TRACE_ON;
            if (*scan++ == ',')
            {
                temp = ListParse (scan);
                stack ->maxdepth = atoi (temp ->string);
                FreeList (temp);
            }
            break;
        }
    }
    free (control);
}


/*
 *      _db_pop_    pop the debug stack
 *
 *      Pops the debug stack, returning the debug state to its
 *      condition prior to the most recent _db_push_ invocation.
 *      Note that the pop will fail if it would remove the last
 *      valid state from the stack.  This prevents user errors
 *      in the push/pop sequence from screwing up the debugger.
 *      Maybe there should be some kind of warning printed if the
 *      user tries to pop too many states.
 */

VOID _db_pop_ ()
{
    REGISTER struct state * discard;

    discard = stack;
    if (discard != NULL && discard ->next_state != NULL)
    {
        stack = discard ->next_state;
        _db_fp_ = stack ->out_file;
        _db_pfp_ = stack ->prof_file;
        if (discard ->keywords != NULL)
            FreeList (discard ->keywords);
        if (discard ->functions != NULL)
            FreeList (discard ->functions);
        if (discard ->processes != NULL)
            FreeList (discard ->processes);
        if (discard ->p_functions != NULL)
            FreeList (discard ->p_functions);
        CloseFile (discard ->out_file);
        CloseFile (discard ->prof_file);
        free ((char *) discard);
    }
}
```

```
/*
 *      _db_enter_    process entry point to user function
 *
 *      VOID _db_enter_ (_func_, _file_, _line_, _sfunc_, _sfile_,
_slevel_)
 *      char *_func_;           points to current function name
 *      char *_file_;           points to current file name
 *      int _line_;             called from source line number
 *      char **_sfunc_;         save previous _func_
 *      char **_sfile_;         save previous _file_
 *      int *_slevel_;          save previous nesting level
 *
 *      Called at the beginning of each user function to tell
 *      the debugger that a new function has been entered.
 *      Note that the pointers to the previous user function
 *      name and previous user file name are stored on the
 *      caller's stack (this is why the ENTER macro must be
 *      the first "executable" code in a function, since it
 *      allocates these storage locations).  The previous nesting
 *      level is also stored on the callers stack for internal
 *      self consistency checks.
 *
 *      Also prints a trace line if tracing is enabled and
 *      increments the current function nesting depth.
 *
 *      Note that this mechanism allows the debugger to know
 *      what the current user function is at all times, without
 *      maintaining an internal stack for the function names.
 *
 */

VOID _db_enter_ (_func_, _file_, _line_, _sfunc_, _sfile_, _slevel_)
char *_func_;
char *_file_;
int _line_;
char **_sfunc_;
char **_sfile_;
int *_slevel_;
{
    if (!init_done)
        _db_push_ ("");
    *_sfunc_ = func;
    *_sfile_ = file;
    func = _func_;
    file = BaseName (_file_);
    stack ->level++;
    *_slevel_ = stack ->level;
    if (DoProfile ())
    {
        (VOID) fprintf (_db_pfp_, "%s\tE\t%ld\n", func, Clock());
        (VOID) fflush (_db_pfp_);
    }
    if (DoTrace ())
    {
        DoPrefix (_line_);
        Indent (stack ->level);
        (VOID) fprintf (_db_fp_, ">%s\n", func);
        (VOID) fflush (_db_fp_);
        (VOID) Delay (stack ->delay);
    }
}


/*
 *      _db_return_    process exit from user function
 *
 *      VOID _db_return_ (_line_, _sfunc_, _sfile_, _slevel_)
 *      int _line_;              current source line number
 *      char **_sfunc_;          where previous _func_ is to be retrieved
 *      char **_sfile_;          where previous _file_ is to be retrieved
 *      int *_slevel_;           where previous level was stashed
 *
 *      Called just before user function executes an explicit or implicit
 *      return.  Prints a trace line if trace is enabled, decrements
 *      the current nesting level, and restores the current function and
 *      file names from the defunct function's stack.
 */

VOID _db_return_ (_line_, _sfunc_, _sfile_, _slevel_)
int _line_;
char **_sfunc_;
char **_sfile_;
int *_slevel_;
{
    if (!init_done)
        _db_push_ ("");
    if (stack ->level != *_slevel_ && (TRACING || DEBUGGING || PROFILING))
        (VOID) fprintf (_db_fp_, ERR_MISSING_RETURN, _db_process_, func);
    else if (DoProfile ())
        (VOID) fprintf (_db_pfp_, "%s\tX\t%ld\n", func, Clock());
    else if (DoTrace ())
    {
        DoPrefix (_line_);
        Indent (stack ->level);
        (VOID) fprintf (_db_fp_, "<%s\n", func);
    }
    (VOID) fflush (_db_fp_);
    (VOID) Delay (stack ->delay);
    stack ->level = *_slevel_ - 1;
    func = *_sfunc_;
    file = *_sfile_;
}
```

The code for the dbug package is continued in the next chapter.

EOF

# Basically OS-9

**Dedicated to the serious OS-9 user.**
**The fastest growing users group world-wide!**
**6809 - 68020**

***A Tutorial Series***

By: Ron Voigts
2024 Baldwin Court
Glendale Heights, IL 60139

## DEVICES

I thought this month we could talk about device descriptors. I am sure your saying what more can be said about them. They are a module used by device managers to interact with some piece of hardware. Well there it is in a nutshell. Its been a fun. See you all next time!

*OK. Ok.* There are a few more things that can be said. ( I thought you would never ask. ) Talking about device descriptors is not the most exciting aspect of OS-9, but it is most necessary. So on with it.

I think it is first necessary to understand the nature of peripherals. Those are the things we attach to the computer and expect it to interact with. These include items like monitors, keyboards, terminals, modems, printers, disk drives, and hard disks. The all have one thing in common. They require a communication link with the computer.

OS-9 is clever. It devised a method to talk to all these in a simple and straight forward fashion. Everything boils down to input and output. To handle this OS-9 has a special program called the Input/Output Manger or just IOMan.

In the simplest sense data flows back and forth. Characters flow to your video monitor and from your keyboard. This type of interaction is know as sequential since everything flows one byte after another. To handle this OS-9 has a program called the Sequential Character File Manger or SCFMan for short.

This method works fine for terminals and printers, but it is not practical for disk drives and hard disks. These devices move data in chunks. This chunk is more technically known as a block which consists of a fixed amount of data bytes. The blocks of information are stored in apparent "random" fashion on the device's storage media. ( The word "random" makes me feel that things are stored in a helter-skelter fashion. If this were the case we would not get very far. ) The program that handles this is called the Random Block File Manager or ( you guessed it! ) RBFMan.

There is also a Pipe File Manager called PIPEMan. It handles pipes. Can you guess what the SBFMan is? Beep! Time is up. It stands for Sequential Block File Manager. This is the type of data that is sent to tape drives.

Below the level of file managers are device drivers. These executable modules are real workers. They are specifically designed to work with a particular device. For every type of device there is a driver. On my Level II, Smoke Signal system, the floppy disk drive controller is their model DCB-4A. The driver is DCB4 ( good name! ). Now on my Level II, Color Computer 3 system, the driver is CC3DISK ( another good name! ). The drivers are not interchangable. They are two separate items. If by chance I got a new board for the Smoke Signal system and it was called XYZ123, the old driver would not work. And consequently the same hold true for the Color Computer System. ( I have got an disk driver for the Color Computer, Level I, that I am partial to, but it won't work with the Level II system. ) So the final word is drivers are specific. Unlike the managers which are geared to a certain class of devices, the driver is very specific to the type of device.

There are many types of devices out there. Many times they travel in groups. Do you have a one or more drives? Can your system handle more than one printer? The answer to these question is most likely, YES. And what about those extra drives or extra printers. Are the drives one or two sided? Are they 48 tpi or 96 tpi? What about the printers? Are they running at the same baud rate? Do they both require line feeds? These questions and more are answered by the device descriptors. ( And that brings us to the this month's discussion. )

I have included an example of a device descriptor in Listing 1. I will have more to say about it a little later. But for now the listing is a good example of how to write a descriptor.

The header is similar to most modules with a few additions. The first 9 bytes consist of items like the sync bytes, module size, name offset, type, language, attributes, revision and header parity. After the parity byte comes the information that makes it a descriptor. It consists of

| BYTE | SIZE | USE |
|---|---|---|
| $09 | 2 | File manager offset |
| $0B | 2 | Device driver offset |
| $0D | 1 | Mode capabilities |
| $0E | 1 | Extended address |
| $0F | 2 | Port address |
| $11 | 1 | Bytes in initialization table |
| $12 | n | n bytes of initialization table |

The initialization table follows. The one appearing in Listing 1 is for a floppy disk on my Color Computer 3, OS-9 Level II.

IT.DTP is the device type.
0...SCF
1...RBF
2...PIPE
3...SBF

IT.DRV is the drive number.
0...drive 0
1...drive 1
2...drive 2
etc.

IT.STP is the step rate. It varies with controller type. For my system the rates are

0...30mS
1...20mS
2...12mS
3....6mS

IT.TYP is the disk type

Bit 0 = 0...5 inch disk
Bit 0 = 1...8 inch disk
Bit 5 = 0...Non-color computer format
Bit 5 = 1...Color Computer format
Bit 6 = 0...Standard OS-9 format
Bit 6 = 1...Non-standard format
Bit 7 = 0...Floppy disk
Bit 7 = 1...Hard Disk

IT.DNS is the media density.
Bit 0 = 0...Single density (FM)
Bit 0 = 1...Double density (MFM)
Bit 1 = 0...Single track (48 TPI)
Bit 1 = 1...Double track (96 TPI)

IT.CYL is the number of cylinders (tracks) per side. 35, 40 and 80 per side are standard.

IT.SID is the the number of sides. It is usually 1 or 2 for floppy disks.

IT.FVY is verify disk writes.
0...verify
1...no verify

IT.SCT is sectors per track. This is usually 16 or 18 for 5 inch disk.

IT.T0S is sectors on track 0. This can vary from IT.SCT. Usual numbers are 10, 16, and 18 for 5 inch disk.

IT.ILV is the sector interleave. This is an interesting number. It is the skip factor for reading or writing a disk. After a sector is read, by the time the information is assimilated a few sectors may have passed. So this tells it how far to skip. I use a factor of $03.

IT.SAS is the minimum number of sectors allocated at one time.

## DEFAULT DRIVE

In most cases, two locations of importance are the working data directory and the execution direction. In many cases these have caused problems. If for example you wanted to hard code where you program would find some special file, you would have to settle on where the default drive would be. This was a problem with many software packages. In my C compiler, it prefers to look at /d1. What if I did not have a /d1? Too bad!

An altemate I tried is change the /d1 in the C compiler to ... ( dot-dot-dot ). This caused the compiler to look two directory levels above the current working directory. The solution was fine, except it still did not allow flexibility. Here is where the default drive comes in. It is named DD, but can be any device. It can be for drive 0 or drive 1 or a hard drive or anywhere.

In Listing 1 is a device descriptor for the Color Computer 3 using drive 0 as DD. In reality the Level II system for the Color Computer comes with a device driver DD, but the one in the Listing is different. This one has a stepping rate of 6 mS, not the 30 mS that is standard with OS-9 from Tandy.

Now it has occurred that there might be some who do not want to rewrite their device descriptors or do not have assemblers. So I created the program in Listing 2. It can accomplish the same thing, but in a different manor. This program takes an existing module and changes its intemal name.

To use this one, a copy of your device descriptor must be saved. Then the program ChgName is run on it to change its name to DD. And finally the new descriptor's CRC is corrected. In terms of keyboard input, the following will work for changing D0.

```
OS9: save d0
OS9: chgname("DD") <d0 ! verify u >dd
```

DD will now exist and be a descriptor for drive 1. Similarily for drive 1 or a hard disk or any RBF device, a default driver can be created.

The way ChgName works is very simple. It can be broken down into steps.

1. Get the header and write it.
2. Read and write up to the name.
3. Write the new name.
4. Skip past the old name.
5. Read and write to end-of-file.

There are some problems that arise when using it to change module names . If a name is changed to something of a different length, it is possible that the new module will not work. By changing the name length, everything after it will be offset by an equivalent amount. In machine code, items that are referenced to the relative position will find that the position has changed. For example, I changed the name of DIR to D and found it crashed. On the otherhand, I changed it to my name, RON, and it worked fine. Use ChgName with discretion and caution.

Another month has come and gone. I leave you with the two programs at the end of column. I have been talking with Don Williams and hopefully in the near future, we will be offering the programs from this column on disk form. More about that later. Have a good month!

**Editor's Note:** *If you are interested in the source code for the programs that have been published, drop us a line so that I can get a fell for about how many copies to prepare.*
*Thanks*

*DMW*

## LISTING 1

```
               ********************************
               *
               * Name: DD
               * By: Ron Voigts
               * Date: 16-APR-88
               * To compile: asm dd.a o=/d0/cmds/dd 1 #20k
               *
               *******************************
               *
               * Version 1.00  Original  RDV
               *
               *****************************
               *
               * Function:
               *    This is device driver used
               *    for a default directory.
               *

                                use   /d0/defs/defsfile
         0001           LEVEL   equ   1  select level one
                                ifp1
                                endc

         FF40           CPort   equ   $FF40

         00F1           TYPE    set   Devic+Objct
         0082           REVS    set
```

```
ReEnt+2

0000 87CD0030            mod    DDEnd,DDName,TYPE,REVS,DDMgr,DDDrv

          * Descriptor header
000D FF                  fcb    %11111111  Device Capabilities
000E 07                  fcb    $07        Extended Address
000F FF40                fdb    CPort      Disk Controler Port
0011 0F                  fcb    DOptEnd-*-1

          * Initialization table
0012 01        IT.DTP    fcb    $01        RBF device
0013 00        IT.DRV    fcb    $00        Device number 0
0014 03        IT.STP    fcb    $03        6mS step rate
0015 20        IT.TYP    fcb    $20        Color Computer, 5" disk
0016 01        IT.DNS    fcb    $01        double densty, 48 tpi
0017 0028      IT.CYL    fdb    $0028      40 tracks/side
0019 02        IT.SID    fcb    $02        2 sides
001A 00        IT.VFY    fcb    $00        Verify disk writes
001B 0012      IT.SCT    fdb    $0012      18 sectors/track
001D 0012      IT.TOS    fdb    $0012      18 sectors/track 0
001F 03        IT.ILV    fcb    $03        Disk interleave of 4
0020 08        IT.SAS    fcb    $08        Allocate 8 sectors opening fil
0021           DOptEnd   equ    *

0021 44C4      DDName    fcs    /DD/       Device name DD
0023 5242C6    DDMgr     fcs    /RBF/      Device manager
0026 43433344  DDDrv     fcs    /CC3Disk/  Device driver

002D 2A46A0              emod
0030           DDEnd     equ    *
                         end
```

LISTING 2

```
PROCEDURE chgname
0000       (* ********************
0017       (*
001A       (* Name: ChgName
002A       (* By: Ron Voigts
003B       (* Date: 23-APR-88
004D       (* To compile:  Pack to CMDS directory
0073       (*
0076       (* ********************
008D       (*
0090       (* Function:
009C       (* This modudle will change an
00BA       (* executable module's name to
00D9       (* any new name.
00E9       (*
00EC       (* ********************
0103       (* Usage:
010C       (* chgname("new_name") <old_name ! verify u >new_name
0141       (* attr new_name e pe
0156       (*
0159       (* ********************
0170       (*
0173       (* External modules:
0187       (* 1. RUNB
0191       (* 2. VERIFY
019D       (* 3. ATTR
01A7       (*
01AA       (* ********************
01C1
01C2      (* New module name
01D4      PARAM new_name:STRING
01DB
01DC      (* Module Header
01EC      TYPE header=id,siz,nam:INTEGER; typ,rev,par:BYTE
020D      DIM m:header
0216
0217      (* Other variables
0229      DIM inpath,outpath:BYTE
0234      inpath:=0
023B      outpath:=1
0242
0243      DIM i:INTEGER
024A      DIM c:BYTE
0251
0252
0253      (* Get module header
0267      GET #inpath,m
0271      PUT #outpath,m
027B
027C      (* Move up to module name
0295      FOR i:=SIZE(m) TO m.nam-1
02AF        GET #inpath,c
02B9        PUT #outpath,c
02C3      NEXT i
02CE
02CF      (* Put in new name
02E1      FOR i:=1 TO LEN(new_name)
02F3        c:=ASC(MID$(new_name,i,1))
0302        IF i=LEN(new_name) THEN
0310          c:=c+$80
031C        ENDIF
031E        PUT #outpath,c
0328      NEXT i
0333
0334
0335      (* Move past old name
034A      LOOP
034C        GET #inpath,c
0356      EXITIF c>=$80 THEN ENDEXIT
0366      ENDLOOP
036A
036B      (* Process the remaider of the module
0390      WHILE NOT(EOF(#inpath)) DO
039B        GET #inpath,c
03A5        PUT #outpath,c
03AF      ENDWHILE
03B3
03B4      END
```

EOF

# Logically Speaking

Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you - *what you want!*

## The Mathematical Design of Digital Control Circuits

By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2

Mile 12 - heading for Mile 13

Well, here we are at the first anniversary of the start of this course. Happy birthday, everyone!! By way of a light introduction to Year 2, let's begin with

### COMPUTATIONAL NETWORKS

I guess you won't be TOO surprised to learn that, besides being able to control a sequential machine, relays can also be made to carry out mathematical computations. Sometimes we need to design such a circuit to demonstrate what can be done with modern design techniques, or as an attention-getter at a high-school "Look what relays can do!" type of exhibition. Examples of this type of circuit are

Given 'n', compute n^2 or 3n - 2 or n^2 - n + 1. Or, given two numbers 'm' and 'n', compute m * n or 4m - n + 1 or m^2 - n^2. And so on.

In order to give you a feel for this design method, we'll develop circuits for both one and two variables. Although this sounds a LOT trickier than some of the circuits we've developed so far, it's actually not as difficult as it sounds, and we'll prove it by starting off with a simple example in a single variable.

### SINGLE-VARIABLE COMPUTATIONS

Assuming we have three pushbuttons labelled X4, X2 and X1 (corresponding to binary 4, 2 and 1), we note that by pressing various combinations (or none at all) of these buttons we can create any number from 0 through 7. The problem is to design a circuit which will square the number set up and then add 1 to it, that is, the circuit is to compute n^2 + 1.

| Z \ n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| L1 | 1 | | 1 | | 1 | | 1 | |
| L2 | | 1 | | 1 | | 1 | | 1 |
| L4 | | | 1 | | | | 1 | |
| L8 | | | | 1 | | 1 | | |
| L16 | | | | | 1 | 1 | | 1 |
| L32 | | | | | | | 1 | 1 |

Diagram 49

Our first step is to calculate the maximum value of the function. In this case 7^2 + 1 = 50, which tells us that to display the result with binary-numbered lights we'll need 6 lights to indicate the binary bit-positions 32, 16, 8, 4, 2 and 1. Accordingly we construct the table of Diagram 49, with the 'n' columns headed 0 through 7, and the rows labelled L1, L2 etc up to L32. The design is carried out by calculating the value of n^2 + 1 for each value of 'n' given in the heading and inserting a '1' in the column below, opposite the appropriate lights. For example, in column 0, n^2 + 1 = 0^2 + 1 = 1, so a '1' is inserted in column 0 opposite L1. Similarly, when n = 6, n^2 + 1 = 6^2 + 1 = 37, and so in column 6 a '1' is inserted opposite L32, L4 and L1. Each column is systematically completed in this manner to give the final table of Diagram 49. This corresponds to the intermediate decoding-table which we'd normally construct from a flow-table.

Now, having designed VERTICALLY, we decode HORIZONTALLY to obtain our control-circuit. What could be simpler? The decodings are shown in Diagram 50. Note that our table in Diagram 49 shows quite clearly that L2 is the complement, or negation, of L1, so we merely complement L1'S decoding to obtain that for L2.

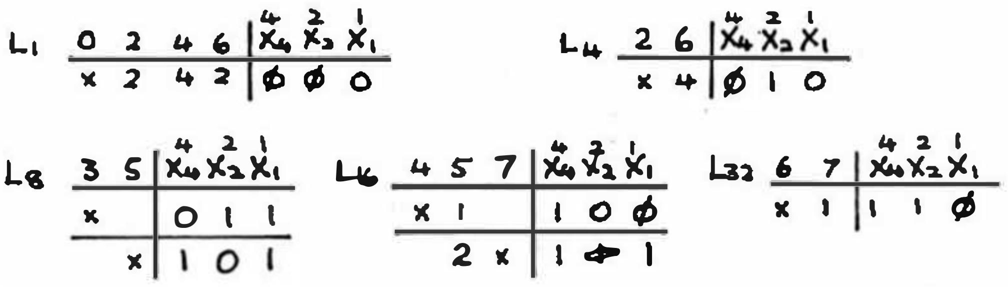


Diagram 50

L8's decoding is unusual in that not one of its variables "goes", and we are left with a 6-literal decoding, which can be reduced to five by factoring out the X1 variable.

The final decodings for the various lights are :

L1 = X1'    L2 = X1
L4 = X2.X1'    L8 = X4'.X2.X1 + X4.X2'.X1 = X1(X4'.X2 + X4.X2')
L16 = X4.X2' + X4.X1 = X4(X2' + X1)    L32 = X4.X2

Of course, if we wished instead to have the result displayed decimally, we would have coded vertically for L1, L2, L3 ... L8, L9, L10, L20, L30, L40 and L50.

MULTI-VARIABLE COMPUTATIONS

Now for a SLIGHTLY more difficult example! Let's assume we have TWO sets of pushbuttons, enabling us to set up values for the number 'm' and the number 'n', and our circuit has to multiply these two numbers together, double the result and then add 1, that is, it must compute 2mn + 1. We'll work this one out with the added restriction that both 'm' and 'n' are limited to the values 0 through 3.
It's obvious that both numbers will require 2 buttons, with the binary values 1 and 2, in order to set up the figures 0 to 3, giving a total of 4 buttons. Four buttons means a total of 4 binary bits, which, in turn, are capable of representing the combined binary range 0 through 15. This tells us that our intermediate-table will require 16 columns, headed 0 to 15, with an additional 2 rows of data above this header - one for 'n', and one for 'm' above that.
The highest number we can compute is when both 'm' and 'n' are at their maximum, that is 3, giving us 2 * 3 * 3 + 1 = 19, for which we're going to need output lights ranging from L1 through L16, or 5 output rows.

| m | 0 | | | | 1 | | | | 2 | | | | 3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| L1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| L2 | | | | | | 1 | | 1 | | | | | | 1 | | 1 |
| L4 | | | | | | | 1 | 1 | | 1 | | 1 | | 1 | 1 | |
| L8 | | | | | | | | | | | 1 | 1 | | | 1 | |
| L16 | | | | | | | | | | | | | | | | 1 |

Diagram 51

Note that we set up the heading for 'n' by repeating the sequence 0 - 3 for each of the values 0 - 3 for 'm'. We have thus paired off all possible combinations of 'm' with all possible combinations of 'n'.
As before, we design vertically, commencing with column 0, where both 'm' and 'n' are equal to 0, so that 2mn + 1 = 1. We record this by by placing a '1' opposite L1 in Column 0, and similarly in the next 3 columns where m = 0. In Column 10, where both 'm' and 'n' equal 2, 2mn + 1 equals 9, which is recorded by placing a '1' opposite L1 and L8 in this column. Once the whole table is completed, we decode horizontally as before, noting, by the way, that the row for L1 is completely filled with 1s. This indicates that L1 is permanently ON from the moment of switching ON our circuit - it's permanently

wired across the power supply.
I'll leave the actual decoding to you as an additional exercise, as the method is quite straightforward (no phis in this table) and the extra practice will not come amiss. Don't skip doing this, though you may use your decoder-cards if you like! I'll just give you one piece of info, and that is that the decoding-table's variable header will look like

```
 8  4 2 1
m2 m1 n1 n2
```

Notice that 'm' comes first, because it occupies the higher level of the intermediate-table's header, and is then followed by 'n' - - with two binary bits apiece.

## CIRCUIT IMPLEMENTATION WITH STANDARD AND-OR-NOT LOGIC UNITS

Normally I wouldn't be dealing with solid-state for a little while to come, as I used to give these lessons on a weekly basis, but as we're already into our 13th month I think I'll take a little time out to show you how to implement AND-OR-NOT logic units in our circuit design. I'll leave NOR and NAND implementation till later, as the decoding technique for these is a LITTLE more complex.
The whole design technique for standard, or AND-OR-NOT, logic is exactly the same as for relay implementation. The one big difference is that we won't need multi-pole pushbuttons, or even buttons with complementary contacts. A simple light-weight button with only a single NO-contact will suffice. We'll use our single-variable example of n^2 + 1 to show how the various logic units are connected.

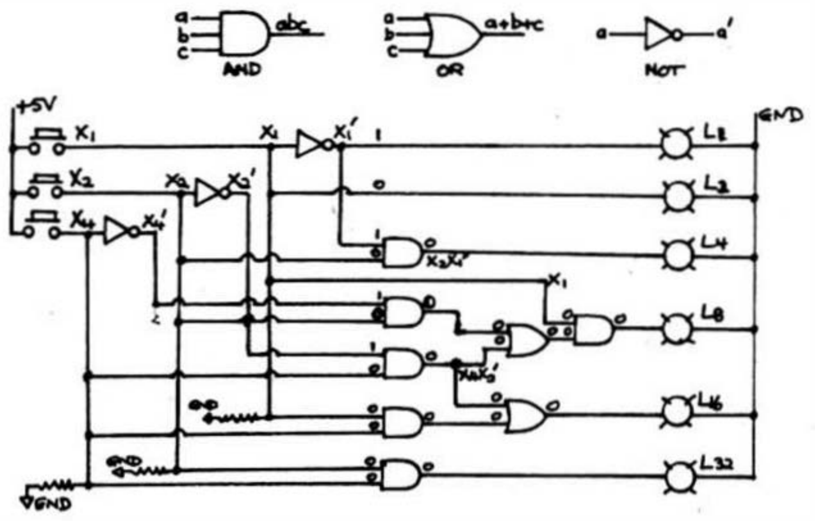


Diagram 52

Along the top of Diagram 52 the logic symbols for the three devices are shown. For the enlightenment of newcomers to this field, I'll explain that an AND unit will output a binary '1' if all its inputs are equal to '1' (that is, input-A AND input-B AND input-C, etc, are all equal to '1'), whereas an OR unit will output a logic '1' if either input-A OR input-B OR input-C, etc, is equal to '1' (OR any combination thereof), otherwise the output is equivalent to logic-0. A NOT unit merely outputs the complement of its input, outputting a '1' if the input is '0', and vice-versa.
It's fairly standard practice to keep the inputs at the left, and to work towards the outputs at the right. We may also use any intermediate function of one of the outputs as part of another output's circuit. For example, the X4.X2' of L8 can be used for L16, though this wouldn't be possible with relays. In addition, as we're now working with low-voltage circuitry, we could use LEDs for our output lights.
And so, back to relays for a while.

## DESIGNING INVERSE NETWORKS

Suppose we have a circuit controlling L1, as shown in Diagram 53a, and we desire the inverse, or complementary, network to control L2. That is to say, we need a network which will guarantee that whenever L1 is ON, L2 will be OFF, and whenever L1 is OFF then L2 will be ON.

There are several ways of designing such a circuit. We could, for instance, merely negate the Boolean expression for the original network, and draw the circuit-diagram for the resultant expression. The present network is expressed as

$$L1 = (a + c')(b' + de') + f$$

and its complement is therefore

$$L2 = [(a + c')(b' + de') + f]' = (a'c + b(d' + e))f$$

Another method would be to enter the expression on a K-map in the form of 1s, and then read out the 0s as if they were 1s. But as we haven't yet learned advanced K-mapping, we can't for the moment comfortably handle more than 4-variable maps. There is another complication, too, in that, to date, we've been dealing only with ladder-type relay networks. That is, we've constructed only series-parallel networks, and the above techniques would be almost useless, perhaps even worthless, in the case of "bridge" or other non-series-parallel networks. However, by using "graphical complementation" all networks, with the exception of non-planar networks, CAN be very easily complemented.
A "non-planar" network, by the way, is one which CANNOT be drawn on a sheet of paper without at least one connecting-line crossing another. Several networks you may come across may LOOK like non-planar ones, but very often the line-crossings can be eliminated by re-arranging the drawing. Only if it's ABSOLUTELY IMPOSSIBLE to do so is the network truly non-planar. We'll be dealing with this type of network in the not-too-distant future in a lot more detail!
Anyway, returning to 53a, the first step in obtaining the inverse network is to place a dot, or "node", inside each closed loop, or "mesh", of the network, as shown in 53b. In addition, an extra node is placed above the network and one below. Now, if it's possible to get from one node to another by passing through ONE contact only, then a dotted line should be drawn connecting these two nodes. All such possibilities are indicated in 53c, and L2 has been added to the lower end. For the sake of clarity, the contacts have not been labelled in this diagram.
If we now re-draw this diagram, as in 53d, AND COMPLEMENT ALL THE CONTACTS, we've drawn the complementary network called for in our specifications. Of course, the contact network of 53d would actually be swung round anti-clock-wise to form the normal right-to-left ladder configuration.

## TRANSFER-CONTACT NETWORKS

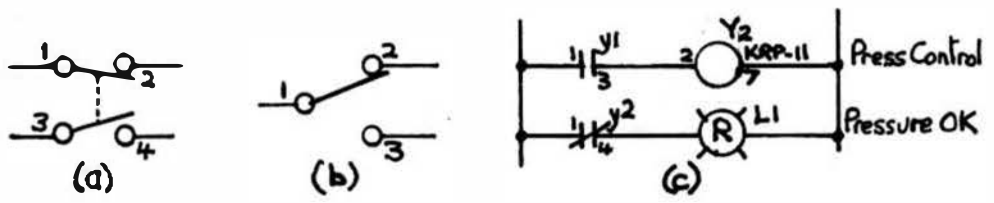
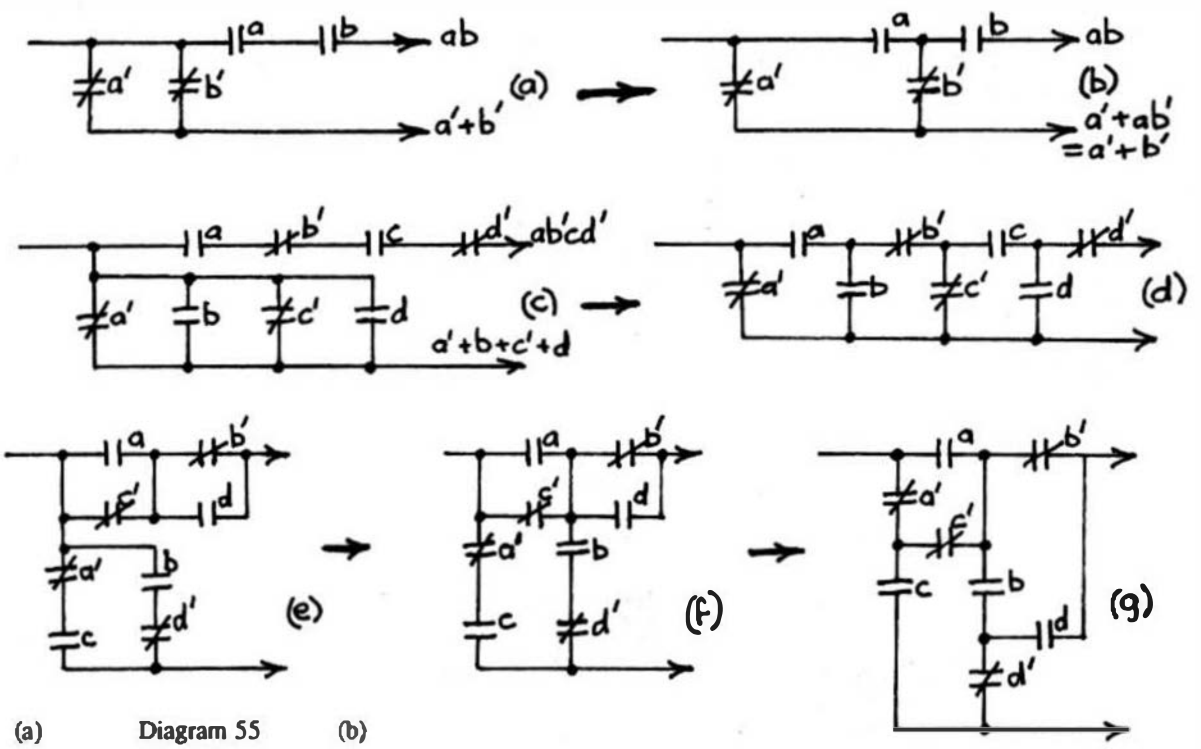


Diagram 54

Relay contacts may come in different configurations, Diagram 54 showing two of the most common. In 54a we see separate NO (normally-open) and NC (normally-closed) contacts, and, of course, there may be several such pairs on a single relay. Diagram 54b shows a combined NO and NC configuration, called a "transfer-contact" because the switch transfers the current-flow from the upper to the lower contact.
Observe the numbers attached to the contacts. It's normal procedure with plug-in type relays to incorporate the relay's pin-numbers into the circuit-diagram, with the left-most number usually above the horizontal line and the right-hand number below, wherever possible. For larger, hard-wired relays we'll designate the various contacts as y1a, y1b, and so on, to ease the burden of trouble-shooting later. Note also in 54c, that the relay-coil not only includes its pin-numbers, but has the relay-type or catalog-number added, together with a brief description of its function. Lights will have an internal letter denoting their colour, "R" for red, "G" for green, etc, plus a description of their function.
Transfer-contacts occur so frequently that it's well worthwhile looking at a few techniques specially directed towards minimising the number of "transfers", as they're usually called, where this type of relay is utilised. Let's see what can be done, assuming that we're using only transfer relays.

## 1. COMBINING A NETWORK AND ITS COMPLEMENT



(a) Diagram 55 (b)

If we have a simple network with two outputs, one of which is the exact complement of the other, as in 55a, the circuit as shown would need 3 transfers - one on relay-A (because the a and the a' contacts have a common point) and two on relay-B, using the NO-contact of one transfer and the NC-contact of the second. However, if we break apart the section of circuit composed of contacts in parallel, and transfer the "loose end" to the appropriate part of the series path, the function will remain unchanged. In 55a we've lifted off the wire joining one end of the b'-contact to the a-contact, and transferred it to the junction of the a-contact and the b-contact in the series path, thus giving both contacts on relay-B a common point too. This move reduces the total number of transfers from 3 to 2, without changing the function of either path.
Two more examples are given in 55c and 55d, and 55e, f and g. In the last example, note how the process of breaking-up and re-arranging the parallel paths is repeated to produce a network which is more difficult to understand, but which, neverthe-less, has reduced the total number of transfers required from seven to four.
Don't forget, this method is used WHERE ONE NETWORK IS THE COMPLEMENT OF THE OTHER.

## 2. COMBINING LIKE CONTACTS IN SEPARATE PARALLEL PATHS

Sometimes it turns out that more CONTACTS will be required in a network in order to reduce the number of TRANSFERS, as we'll see next.

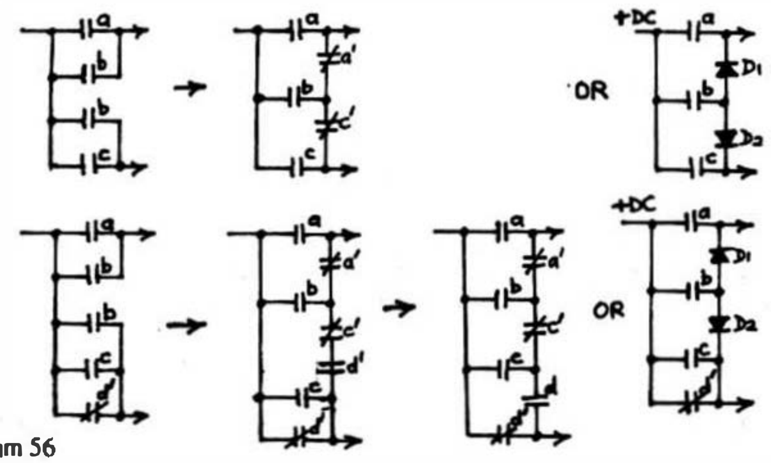


Diagram 56

Diagram 56 shows how the b-contact (which really symbolises ANY network common to the two outputs) need be used only once, by inserting the complement of the other contact into the network. Each of the original parallel paths is thus transformed from the form a + b into its equivalent form a + a'b (remember Rule 6 of The Laws of Boolean Algebra?).
The second example demonstrates the use of both this technique and the previous one to obtain a minimum transfer realisation of the initial network.
In both these cases we end up with MORE contacts, but a lesser number of transfers, which does represent an actual saving, as the complementary contacts were already available AND UNUSED on the relays concerned.
Another technique is to use a DC source of power (instead of AC) as we've assumed till now). Then, by utilising DC relays, a couple of diodes will do the trick instead of figuring out a re-arranged contact network. The diodes, of course, serve to isolate the outputs so that the remainder of one's network will not activate the other output.
If, as sometimes happens, the parallel paths contain a contact and its complement in addition to the common like contact (or block of contacts), the common like contacts can be shifted so that a bridge is formed between the two outputs, with no change in either of the functions.

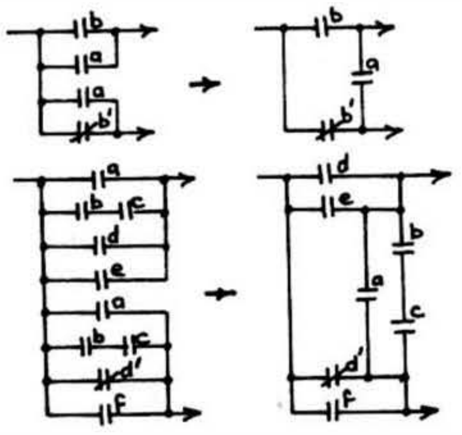


Diagram 57

Diagram 57 illustrates the method for two different examples, and shows how a common block, no matter how large, need be constructed only once PROVIDED ONE OF THE OUTPUT PATHS CONTAINS THE COMPLEMENT OF ONE BRANCH OF THE OTHER OUTPUT PATH. Again you should note that although only a single complementary contact is shown in each example, it's symbolic of ANY path, no matter how complicated.
Diagram 58 gives another example of how it's possible to use more than one of these techniques to re-arrange a network.

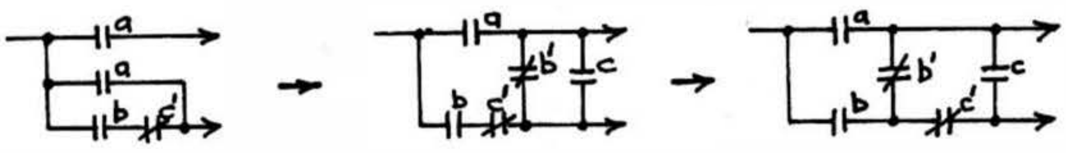


Diagram 58

## RELAY TREES

A relay tree, or transfer-tree, has a single power input and any number of distinctly separate outputs, such that each output path passes through one contact on each relay, and only one output is energised at any one time. Such a tree is shown in Diagram 59, where each of the eight separate outputs is energised through one contact on each of the three relays A, B and C. The fact that only one output is energised at any one time is easily verified by converting the Boolean expression for each path into binary, and thence into decimal if need be. In our diagram, the outputs represent the binary numbers 000 to 111 inclusively, or the decimal numbers 0 to 7.

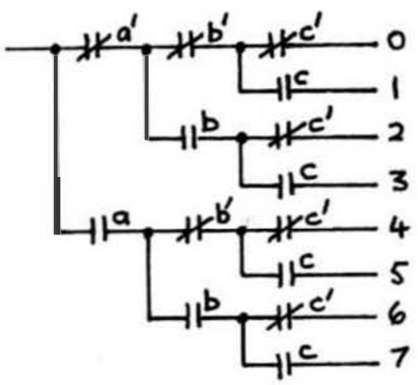


Diagram 59

Notice that while only one transfer-contact is used on relay A, two are necessary on relay-B and four on relay-C. This, of course, is the binary sequence 1, 2, 4 ..... and means that if we had a 6-relay tree the sixth relay would require 32 transfer contacts. If the number of relays is represented by 'n', the number of transfer-contacts necessary on the nth relay would be 2^(n-1). That is, in the case of six relays it would be 2^5, or 32.
A full relay-tree has an output for each and every possible relay combination, and is composed of a total of 2^n - 1 transfers (NOT the same as our previous formula). In the case of a 6-relay tree this works out to 2^6 - 1 = 64 - 1 = 63 transfer-contacts altogether!
Returning to Diagram 59, the transfer-contact distribution is

A B C
1 2 4 = 7 transfers

Although it's not possible to reduce the TOTAL number of transfers in a FULL relay-tree, the distribution of the contact-load CAN be equalised by re-arranging the tree, apart from the first relay connected to the power-input - which has only one transfer.

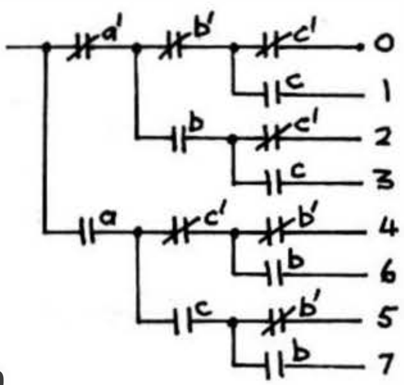


Diagram 60

Diagram 60 shows how the remaining 6 transfers are divided equally between relays B and C to use only 3 transfers apiece. The distribution is now

A B C
1 3 3 = 7 transfers

If the relay-tree is reversed from left-to-right so that we have any number of distinct inputs connected to only one output, the network is known as a reverse tree.

**PARTIAL TREES**

Now, while the total number of transfers cannot be reduced in a FULL relay-tree, it's possible to do so in the case of a PARTIAL tree by re-arranging the tree. A partial tree is one in which some of the "branches" are missing. Moreover, there's a technique, which uses a K-map in a completely different way from heretofore, whereby we can determine the BEST method of re-arranging the transfers.

Let's look at an example to see how it's done. Suppose we're called on to minimise the following output combinations in relay-tree form, keeping in mind that each term represents a SEPARATE output

a'b'c'd' a'b'c'd a'bc'd' a'bc'd ab'cd' ab'cd abc'd' abcd'

Normally, a K-map is used to record the function for only one output, but in the case of relay-trees each 1-entry corresponds to a single output of the tree, AND MUST THEREFORE BE READ AS A SINGLE ENTRY. There must be no forming of loops, as this would be equivalent to bundling a set of independent output functions as though they were one.

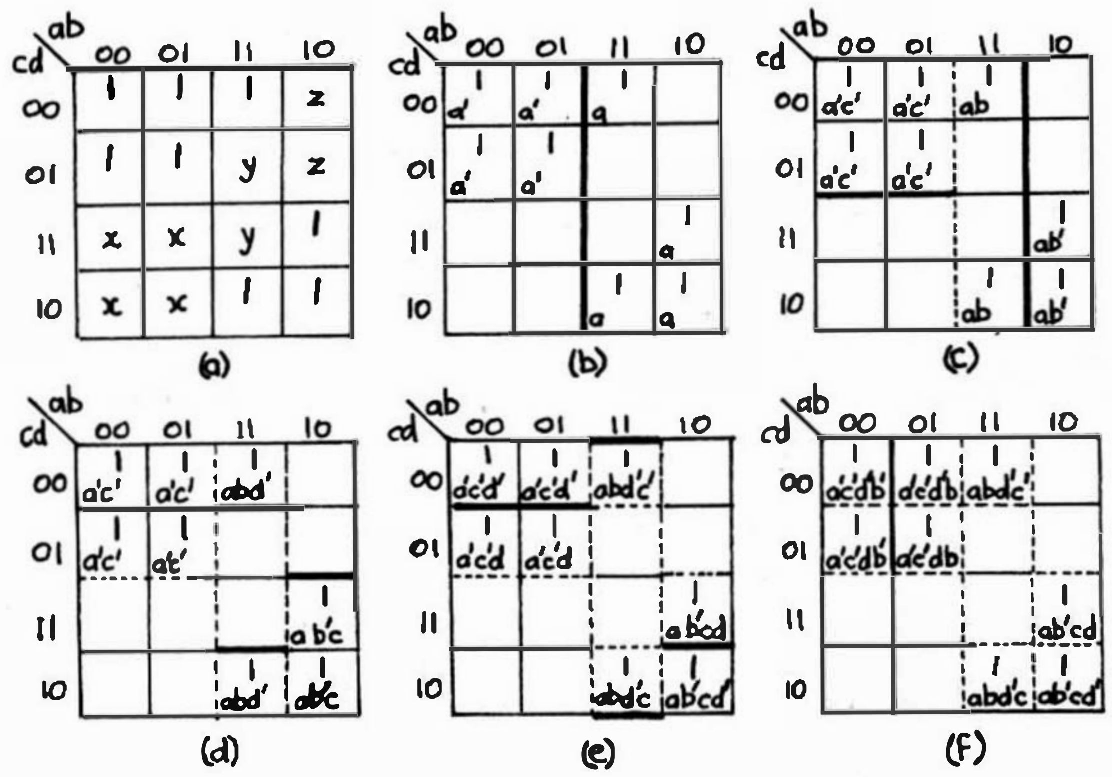


Diagram 61

Our first step, Diagram 61a, is to enter 1s on the K-map to correspond with each of the required outputs. The design process consists of cutting the K-map into two EQUAL parts, each of which is called a "sub-map", and then to cut these sub-maps into two EQUAL parts also, to form new sub-maps, in such a way that the UNUSED squares of the K-map form AS FEW and AS LARGE groups as possible.

**DISSECTING THE K-MAP**

For example, we could cut map 61a into two with a vertical slice down the middle, and then by cutting the left-hand submap horizontally into two we'd isolate the unused squares marked with 'x'. Next we'd cut our right-hand sub-map into two submaps by a vertical slice, and a further horizontal slice in the extreme right-hand sub-map would isolate the block marked 'z'. Finally, and this is a little trickier, we recall that that the top and bottom of a K-map are really adjacent, and so, by imagining our original column 11 as a continuous band, we can make two quick snips horizontally to isolate the block 'y' from the two 1s, which would still be joined together. We'd thus end up with three blocks of unused contacts, so this will be our approach. Of course, we don't ACTUALLY cut the map, but mark it, maybe in red, where our cuts are going to made!

Suppose we were to make our first cut horizontally instead of vertically as we did above! You can see that it would go right through the middle of the two ys, and the best that we could hope for after that would be FOUR blocks of unused squares, consisting of our original 'x' and 'z' and two small blocks of 'y'.
Following our first plan then, we'll make a vertical cut down the centre of the K-map, shown by the heavier line in 61b. If we look carefully at the column headings we see that what we've done is to separate all squares which contain a' as a component from those which contain a . In other words, we've created a a'-submap and an a-submap, and we record this fact by making an appropriate entry in each of the 1-squares, as shown in the diagram.
Proceeding to 61c, we now make a horizontal cut in the a'-submap, and record in its 1-squares the fact that we've thereby separated the cs from the c's. Still with 61c, we'll cut the a-submap vertically into two, forming an ab-submap and an ab'-submap. This, too, is recorded in the corresponding 1-squares.
Moving now to 61d, we make two snips in the "band" forming the ab-submap (remember the 1s are adjacent, and therefore joined), thus isolating an unused block and forming an abd'-submap of 1s. We'll also chop the ab'-submap into two, to give a final isolated block of unused squares and an ab'c-submap of 1s.
Having reached the stage where ALL the unused blocks have been isolated, the last step of the procedure is to concentrate on the submaps containing 1-entries, and to chop them up by the same technique as before until each 1-square is a complete submap in its own right. Diagrams 61e and 61f show these final stages, with the literals being recorded in the various squares as the splitting process is carried out.

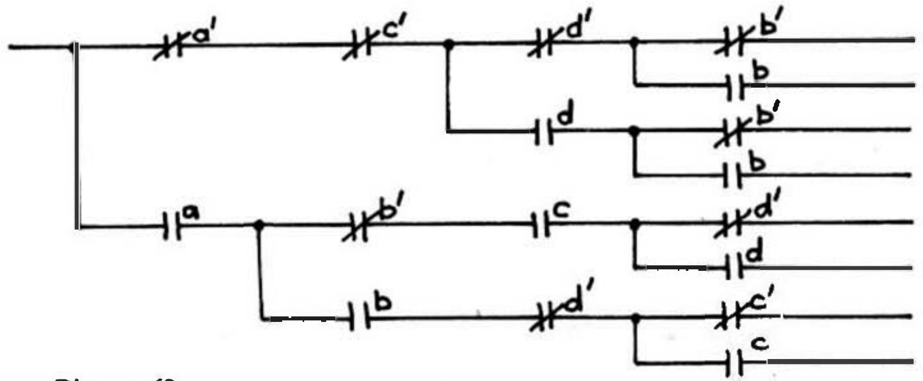


Diagram 62

The relay-tree is constructed from the K-map of 61f, as shown in Diagram 62. The method is to take the 1-squares of the K-map one at a time, commencing in the top-left corner, that is, square 0000. A path is then drawn from power-input to output containing contacts in the EXACT order shown in this square, ie, a' c' d' and b'. The square to the right (0100) has the entry a' c' d' b, but as the first three contacts have already been drawn we merely branch off with a b-contact from the junction of d' and b'. Similarly, when we come to square 0001, we find that we already have the a'c' part, so we branch off into the db' section. And so on for the remaining squares, which gives us a transfer distribution of

A B C D
1 3 3 3 = 10 transfers

And here is where I think we should take a little break, so as not to monopolise TOO much of this magazine, and as you've had things fairly easy of late I'm going to give you another test to sharpen your skills. Here then is

TEST TEN

1. Design the following computational networks, assuming 'n' to range over the values 0 - 7

   (a) n^2 (b) 3n + 1 (c) 2 * n^2 - n + 1

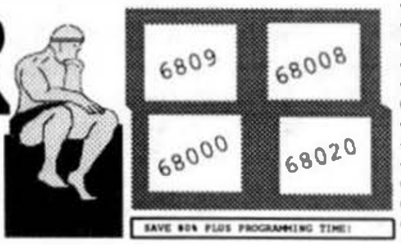
2. Design the following computational networks, assuming 'm' to range over the values 0 - 3, and 'n' over the values 0 - 5

   (a) 2m + n (b) 3m + 2n (c) 3mn + 2

3. Complement the following networks using the graphical complementation technique

*Continued On Page 36*

# SCULPTOR



**From the world's oldest & largest OS-9 software house!**

**CUTS PROGRAMMING TIME UP TO 80%**

**6809/68000-68030 Save 70%**

SCULPTOR-a 4GL - Only from S.E. Media at these prices. OS-9 levels one and two (three GIMIX) 6809, all 68XXX OS-9 standard systems. Regular SCULPTOR versions 1.4:6. One of if not the most efficient and easy to develope DBMS type systems running under OS-9! A system of flexible keyed file access that allows extremely fast record and data retrieval, insertion and deletion or other programmed modifications. Access by key or in ascending order, very fast. The system provides automatic menu generation, compilation and report generation. Practically unlimited custom input format and report formatting. A rich set of maintenance and repair utilities. An extremely efficient development environment that cuts most programming approximately 80% in development and debugging! Portable, at source level, to MS-DOS, UNIX and many other languages and systems.

| | |
|---|---|
| Standard Version: 1.6 | 6809 - $1295.00 |
| | 68000 $1295.00 |
| | 68020 $1990.00 |

**Due to a "Special One Time" Purchase, We Are Making This Savings Offer. Quantities Limited!**

***Once this supply is gone - the price goes back up!***

**System OS-9: 6809/68000-68030**

• Regular ~~$1295.00~~

+ $7.50 S&H USA
Overseas - Shipped Air Mail Collect

ONLY **$295.00**

**S.E. MEDIA**
POB 849
5900 Cassandra Smith Road
HIXSON, TN 37343 615 842-4601

**$AVE - WHILE SUPPLIES LAST!**

*Telephone: (615) 842-4600* **South East Media** *Telex: 5106006630*

*OS-9, UniFLEX, FLEX, SK*DOS*

*SOFTWARE*

*!!! Please Specify Your Operating System and Disk Size !!!*

# SCULPTOR

**Full OEM & Dealer Discounts Available!**

## THE SCULPTOR SYSTEM

Sculptor combines a powerful fourth generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

## AN ESTABLISHED LEADER

Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever-increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

## SYSTEM INDEPENDENCE

Sculptor is available on many different machines and for most operating systems, including MS DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user micros up to large minis and mainframes. Sculptor is constantly being ported to new systems.

## APPLICATION PORTABILITY

Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and – without any alterations to the programs – run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high-speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

## SPEED AND EFFICIENCY

Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

## INTERNATIONALLY ACCEPTED

By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australasia, the Americas and Europe – Sculptor is already at work in over 20 countries.

## THE PACKAGE

With every development system you receive:

- ☐ A manual that makes sense
- ☐ A periodic newsletter
- ☐ Screen form language
- ☐ Report generator
- ☐ Menu system
- ☐ Query facility
- ☐ Set of utility programs
- ☐ Sample programs

For resale products, the run-time system is available at a nominal cost.

**Facts** **Features**

## DATA DICTIONARY

Each file may have one or more record types described. Fields may have a name, heading, type, size, format and validation list. Field type may be chosen from:

- ☐ alphanumeric
- ☐ integer
- ☐ floating point
- ☐ money
- ☐ date

## DATA FILE STRUCTURE

- ☐ Packed, fixed-length records
- ☐ Money stored in lowest currency unit
- ☐ Dates stored as integer day numbers

## INDEXING TECHNIQUE

Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

## INPUT DATA VALIDATION

Input data may be validated at three levels:

- ☐ automatic by field type
- ☐ validation list in data dictionary
- ☐ programmer coded logic

## ARITHMETIC OPERATORS

| | |
|---|---|
| – | Unary minus |
| * | Multiplication |
| / | Division |
| % | Remainder |
| + | Addition |
| – | Subtraction |

## RELATIONAL OPERATORS

| | |
|---|---|
| = | Equal to |
| < | Less than |
| > | Greater than |
| < = | Less than or equal to |
| > = | Greater than or equal to |
| < > | Not equal to |
| and | Logical and |
| or | Logical or |
| ct | Contains |
| bw | Begins with |

## SPECIAL FEATURES

- ☐ Full date arithmetic
- ☐ Echo suppression for passwords
- ☐ Terminal and printer independence
- ☐ Parameter passing to sub-programs
- ☐ User definable date format

**Sculptor for 68020 OS-9 & UniFLEX $995**

## MAXIMA AND MINIMA

| | |
|---|---|
| Minimum key length | 1 byte |
| Maximum key length | 160 bytes |
| Minimum record length | 3 bytes |
| Maximum record length | 32767 bytes |
| Maximum fields per record | 32767 |
| Maximum records per file | 16 million |
| Maximum files per program | 16 |
| Maximum open files | Operating system limit |

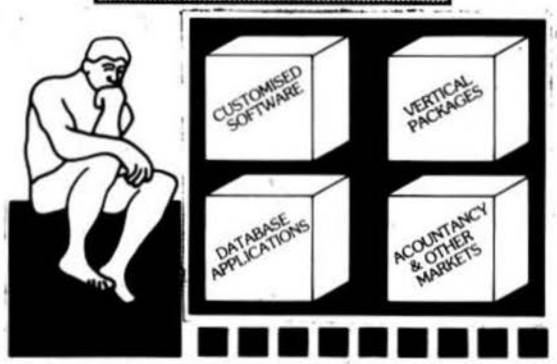
## PROGRAMS

- ☐ Define record layout
- ☐ Create new indexed file
- ☐ Generate standard screen-form program
- ☐ Generate standard report program
- ☐ Compile screen-form program
- ☐ Compile report program
- ☐ Screen-form program interpreter
- ☐ Report program interpreter
- ☐ Menu interpreter

- ☐ Query facility
- ☐ Reformat file
- ☐ Check file integrity
- ☐ Rebuild index
- ☐ Alter language and date format
- ☐ Setup terminal characteristics
- ☐ Setup printer characteristics

## SCREEN-FORM LANGUAGE

- ☐ Programmer defined options and logic
- ☐ Multiple files open in one program
- ☐ Default or programmer processing of exception conditions
- ☐ Powerful verbs for input, display and file access
- ☐ Simultaneous display of multiple records
- ☐ Facility to call sub-programs and operating system commands
- ☐ Conditional statements
- ☐ Subroutines
- ☐ Independent of terminal type



**MUSTANG-020 Users - Ask For Your Special Discount!**

| | * | ** | *** | | * | ** | *** |
|---|---|---|---|---|---|---|---|
| **MUSTANG-020** | *$1,990 | $398 | $795 | **PC/XT/AT/MSDOS** | $695 | $139 | $299 |
| **MUSTANG-08** | *$1,295 | $259 | $495 | **Call or write for prices on the following systems.** | | | |

XENIX SYS III & V, MS-NET, UNIX SYS III & V, ATARI OS-9, 68K, UNOS, ULTRIX/VMS (VAX,REGAL), STRIDE, ALTOS, APRICORT, ARETE, ARMSTRONG, GLEASDALE, CHARLES RIVERS, GMX, CONVERG.TECH, DEC, CIFER, EQUINOX, GOULD, HP, HONEYWELL, IBM, INTEL, MEGADATA, MOTOROLA, NCR, NIXDORF, N.STAR, OLIVETTI/AT&T, ICL, PERKINS ELMER, PHILLIPS, PIXEL, PLESSEY, PLEXUS, POSITRON, PRIME, SEQUENT, SIEMENS, SWTPC, SYSTIME, TANDY, TORCH, UNISYS, ZYLOG, ETC.

*** For SPECIAL LOW SCULPTOR prices especially for 6809/68XXX OS-9 Systems - See Special Ad this issue. Remember, "When they are gone the price goes back up as above!"**

**... Sculptor Will Run On Over 100 Other Types of Machines ...**

**... Call for Pricing ...**

**!!! Please Specify Your Make of Computer and Operating System !!!**

* Full Developement Package
** Run Time Only
*** C Key File Library

**Availability Legends**
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC9 = Color Computer OS-9
CCF = Color Computer FLEX

VISA

*South East Media*
*5900 Cassandra Smith Rd. · Hixson, Tn. 37343*
*Telephone: (615) 842-4600 Telex: 5106006630*

mastercard

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

## *ASSEMBLERS*

ASTRUK09 from S.E. Media -- A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.
*F, S, CCF - $99.95*

Macro Assembler for TSC -- The FLEX, SK*DOS STANDARD Assembler.
*Special -- CCF $35.00; F, S $50.00*

OSM Extended 6809 Macro Assembler from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. Generate OS-9 Memory modules under FLEX, SK*DOS.
*FLEX, SK*DOS, CCF, OS-9 $99.00*

**Relocating Assembler/Linking Loader** from TSC. -- Use with many of the C and Pascal Compilers.
*F, S, CCF $150.00*

MACE, by Graham Trott from Windrush Micro Systems -- Co-Resident Editor and Assembler; fast interactive A.L. Programming for small to medium-sized Programs.
*F, S, CCF - $75.00*

XMACE -- MACE w/Cross Assembler for 6800/1/2/3/8
*F, S, CCF - $98.00*

## *DISASSEMBLERS*

SUPER SLEUTH from Computer Systems Consultants Interactive Disassembler; extremely *POWERFUL!* Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.
*Color Computer SS-50 Bus (all w/ A.L. Source)*
*CCD (32K Req'd) Obj. Only $49.00*
*F, S, $99.00 - CCF, Obj. Only $50.00 U, $100.00*
*CCF, w/Source $99.00 O, $101.00 - CCO, Obj. Only $50.00*
*OS9 68K Obj. $100.00 w/Source $200.00*
*68010 Disassembler $100.00 F, S, O, U, UNIX, MS-DOS*

DYNAMITE+ -- Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options w/ OS-9 Version.
*CCF, Obj. Only $100.00 - CCO, Obj. $ 59.95*
*F, S, " " $100.00 - O, object only $150.00*
*U, " " $300.00*

## *CROSS ASSEMBLERS*

CROSS ASSEMBLERS from Computer Systems Consultants -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/ 146805, 6809/ 00/01, 6502 family, 8080/5, 8020/1/2/35/C35/39/ 40/48/C48/49/C49/50/ 8748/49, 8031/51/8751, and 68000 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text.
*68000 or 6809, F, S, CCF, O, U, *Macintosh, *Atari, UNIX, MS-DOS*
*any object or source each - $50.00*
*any 3 object or source each - $100.00*
*Set of ALL object $200.00 - w/source $500.00*

XASM Cross Assemblers for FLEX, SK*DOS from S.E. MEDIA -- This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format, Assembler options, etc., in providing code for the target CPU's.
*Complete set, FLEX, SK*DOS only - $150.00*

CRASMB from LLOYD I/O -- Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties); 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS9 (binary), and FLEX, SK*DOS (binary). Written in Assembler ... e.g. <u>Very Fast.</u>

**CPU TYPE - Price each:**

| For: | MOTOROLA | INTEL | OTHER | COMPLETE SET |
|---|---|---|---|---|
| FLEX9 | $150 | $150 | $150 | $399 |
| SK*DOS | $150 | $150 | $150 | $399 |
| OS9/6809 | $150 | $150 | $150 | $399 |
| OS9/68K | ------ | ------ | ------ | $432 |

CRASMB 16.32 from LLOYD I/O -- Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS9/ 68K on your OS9/6809 computer.
*FLEX, SK*DOS, CCF, OS-9/6809 $249.00*

## *COMMUNICATIONS*

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".
*FLEX, SK*DOS, CCF, OS-9, UniFLEX, UNIX, MS-DOS, 68000 & 6809 with Source $100.00 - without Source $50.00*

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary files may be directly transferred between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.
*X-TALK Complete (cable, 2 disks) $99.95*
*X-TALK Software (2 disks only) $69.95*
*X-TALK with CMODEM Source $149.95*

XDATA from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.
*U - $299.99*

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

**South East Media**
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

## PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems -- By Graham Trott. A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. 500+ page Manual with tutorial guide.
*F, S, CCF - $198.00*

PASC from S.E. Media - A FLEX9, SK*DOS Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package comes complete with source (written in PASC) and documentation.
*FLEX, SK*DOS $95.00*

WHIMSICAL from S.E. MEDIA Now supports *Real Numbers*. "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. Run-Time subroutines inserted as called during compilation. *Normally produces 10% less code than PL/9.*
*F, S and CCF - $195.00*

KANSAS CITY BASIC from S.E. Media - *Basic for Color Computer OS-9* with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFT$, RIGHT$, MID$, STRING$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.
*CoCo OS-9 $39.95*

C Compiler from Windrush Micro Systems by James McCosh. Full C for FLEX, SK*DOS except bit-fields, including an Assembler. *Requires the TSC Relocating Assembler if user desires to implement his own Libraries.*
*F, S and CCF - $295.00*

C Compiler from Introl -- Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.
*FLEX, SK*DOS, CCF, OS-9 (Level II ONLY), U - $575.00*

PASCAL Compiler from Lucidata -- ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.
*F, S and CCF 5" - $190.00 F, S 8" - $205.00*

OmegaSoft PASCAL from Certified Software -- Extended Pascal for systems and real-time programming.
Native 68000/68020 Compiler, $575 for base package, options available. For OS-9/68000 and PDOS host system.
6809 Cross Compiler (OS-9/68000 host) $700 for complete package.

KBASIC - from S.E. MEDIA -- A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.
*FLEX, SK*DOS, CCF, OS-9 Compiler /Assembler $99.00*

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSII Level 1 *COBOL* with many of the useful Level 2 features. Full FLEX, SK*DOS File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. *A very popular product.*
*FLEX, SK*DOS, CCF - $99.95*

FORTH from Stearns Electronics -- A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!
*Color Computer ONLY - $58.95*

FORTHBUILDER is a stand-alone target compiler (crosscompiler) for producing custom Forth systems and application programs.
All of the 83-standard defining words and control structures are recognized by FORTHBUILDER.
FORTHBUILDER is designed to behave as much as possible like a resident Forth interpreter/compiler, so that most of the established techniques for writing Forth code can be used without change.
Like compilers for other languages, FORTHBUILDER can operate in "batch mode".
The compiler recognizes and emulates target names defined by CONSTANT or VARIABLE and is readily extended with "compile-time" definitions to emulate specific target words.
FORTHBUILDER is supplied as an executable command file configured for a specific host system and target processor. Object code produced from the accompanying model source code is royalty-free to licensed users.
F, CCF, S - $99.95

## EDITORS & WORD PROCESSING

JUST from S.E. Media -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.
* Now supplied as a two disk set:
*Disk #1: JUST2.CMD object file,*
*JUST2.TXT PL9 source:FLEX, SK*DOS - CC*
*Disk #2: JUSTSC object and source in C:*
*FLEX, SK*DOS - OS9 - CC*
The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

**South East Media**
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

MasterCard

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p .u .y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!

*Disk (1) - PL9 FLEX only- F, S & CCF - $49.95*

*Disk Set (2) - F, S & CCF & OS9 (C version) - $69.95*

*OS-9 68K000 complete with Source - $79.95*

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.

*Regular FLEX, SK*DOS $129.50*

*• SPECIAL INTRODUCTION OFFER • $79.95*

*SPECIAL PAT/JUST COMBO (w/source)*

*FLEX, SK*DOS $99.95*

*OS-9 68K Version $229.00*

*SPECIAL PAT/JUST COMBO 68K $249.00*

Note: *JUST in "C" source available for OS-9*

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.

*FLEX, SK*DOS $69.95*

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entire lines, etc. Complete with over 25 different CRT terminal configuration overlays.

*FLEX, CCF, SK*DOS $39.95*

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automaticslly), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

*6800 or 6809 FLEX, SK*DOS or SSB DOS, OS-9 - $175.00*

SPELLB "Computer Dictionary" from S.E. Media -- OVER 150,000 words! *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX, SK*DOS UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

*F, S and CCF - $129.95*

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/SK*DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.

*NEW PRICES 6809 CCF and CCO - $99.95,*

*F, S or O - $179.95, U - $299.95*

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.

*NEW PRICES 6809 CCF and CCO - $69.95,*

*F, S or O - $99.95, U - $149.95*

STYLO-MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.

*NEW PRICES 6809 CCF and CCO - $59.95,*

*F, S or O - $79.95, U - $129.95*

STYLO-PAK --- Graph + Spell + Merge Package Deal!!!

*F, S or O - $329.95, U - $549.95*

*O, 68000 $695.00*

## *DATABASE ACCOUNTING*

**XDMS from Westchester Applied Business Systems**
**FOR 6809 FLEX-SK*DOS(5/8")**

Up to 32 groups/fields per record! Up to 12 character file names! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

**South East Media**
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

ITS EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...)

The possibilities are unlimited...

**FOR 6809 FLEX-SK*DOS(5/8")** **$249.95**

## *UTILITIES*

Basic09 XRef from S.E. Media -- This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.
*O & CCO obj. only -- $39.95; w/ Source - $79.95*

BTree Routines - Complete set of routines to allow simple implementation of keyed files - *for your programs* - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.
*O & CCO obj. only - $89.95*

Lucidata PASCAL UTILITIES (Requires Pascal ver 3)

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.
*F, S, CCF --- EACH 5" - $40.00, 8" - $50.00*

DUB from S.E. Media -- A UniFLEX BASIC decompiler Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works w/ ALL Versions of 6809 UniFLEX basic.
*U - $219.95*

LOW COST PROGRAM KITS from Southeast Media The following kits are available for FLEX, SK*DOS on either 5" or 8" Disk.

1. **BASIC TOOL-CHEST $29.95**
   BLISTER.CMD: pretty printer
   LINEXREF.BAS: line cross-referencer
   REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code
   STRIP.BAS: superfluous line-numbers stripper
2. **FLEX, SK*DOS UTILITIES KIT $39.99**
   CATS. CMD: alphabetically-sorted directory listing
   CATD.CMD: date-sorted directory listing
   COPYSORT.CMD: file copy, alphabetically
   COPYDATE.CMD: file copy, by date-order
   FILEDATE.CMD: change file creation date
   INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents
   RELINK.CMD (& RELINK82): re-orders fragmented free chain
   RESQ.CMD: undeletes (recovers) a deleted file
   SECTORS.CMD: show sector order in free chain
   XL.CMD: super text lister
3. **ASSEMBLERS/DISASSEMBLERS UTILITIES $39.95**
   LINEFEED.CMD: 'modularise' disassembler output
   MATH.CMD: decimal, hex, binary, octal conversions & tables
   SKIP.CMD: column stripper
4. **WORD - PROCESSOR SUPPORT UTILITIES $49.95**
   FULLSTOP.CMD: checks for capitalization
   BSTYCIT.BAS (.BAC): Stylo to dot-matrix printer
   NECPRINT.CMD: Stylo to dot-matrix printer filter code
5. **UTILITIES FOR INDEXING $49.95**
   MENU.BAS: selects required program from list below
   INDEX.BAC: word index
   PHRASES.BAC: phrase index
   CONTENT.BAC: table of contents
   INDXSORT.BAC: fast alphabetic sort routine
   FORMATER.BAC: produces a 2-column formatted index
   APPEND.BAC: append any number of files
   CHAR.BIN: line reader

BASIC09 TOOLS consist of 21 subroutines for Basic09. 6 were written in C Language and the remainder in assembly. All the routines are compiled down to native machine code which makes them fast and compact.

1. CFILL -- fills a string with characters
2. DPEEK -- Double peek
3. DPOKE -- Double poke
4. FPOS -- Current file position
5. FSIZE -- File size
6. FTRIM -- removes leading spaces from a string
7. GETPR -- returns the current process ID
8. GETOPT -- gets 32 byte option section
9. GETUSR -- gets the user ID
10. GTIME -- gets the time
11. INSERT -- insert a string into another
12. LOWER -- converts a string into lowercase
13. READY -- Checks for available input
14. SETPRIOR -- changes a process priority
15. SETUSR -- changes the user ID
16. SETOPT -- set 32 byte option packet
17. STIME -- sets the time
18. SPACE -- adds spaces to a string
19. SWAP -- swaps any two variables
20. SYSCALL -- system call
21. UPPER -- converts a string to uppercase

For OS-9 - $44.95 - Includes Source Code
**Limited Special - $19.95**

**SOFTOOLS**

The following programs are included in object form for immediate application. PL/9 source code available for customization.

READ-ME Complete instructions for initial set-up and operation. Can even be printed out with the included text processor.

CONFIG one time system configuration.

CHANGE changes words, characters, etc. globally to any text type file.

CLEANTXT converts text files to standard FLEX, SK*DOS files.

COMMON compare two text files and reports differences.

COMPARE another check file that reports mis-matched lines.

CONCAT similar to FLEX, SK*DOS append but can also list files to screen.

DOCUMENT for PL/9 source files. Very useful in examining parameter passing aspects of procedures.

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

**South East Media**
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

**** Shipping ****
**Add 2% U.S.A. (min. $2.50)**
**Foreign Surface Add 5%**
**Foreign Airmail Add 10%**
**Or C.O.D. Shipping Only**

***OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.**

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

**ECHO** echos to either screen or file.
**FIND** an improved *find* command with "*pattern*" matching and wildcards. Very useful.
**HEX** dumps files in both hex and ASCII.
**INCLUDE** a file copy program that will accept "includes" of other disk files.
**KWIC** allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.
**LISTDIR** a directory listing program. Not super, but better than CAT.
**MEMSORT** a high-speed text file sorter. Up to 10 fields may be sorted. Very fast. Very useful.
**MULTICOL** width of page, number of columns may be specified. A MUST!
**PAGE** similar to LIST but allows for a page header, page width and depth. Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.
**REMOVE** a fast file deleter. Careful, no prompts issued. Zap, and its gone!
**SCREEN** a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.
**SORT** a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on n* word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.
**TPROC** a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.
**TRANSLIT** sorts a file by x keyfields. Checks for duplications. Up to 10 key files may be used.
**UNROTATE** used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.
**WC** a word count utility. Can count words, characters or lines.
**NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, w/ source (PL9). 3 5-1/4" disks or 1 8" disk w/o source.**
**Complete set SPECIAL INTRO PRICE:**
5-1/4" w/source FLEX - SK*DOS - $129.95
w/o source - $79.95
8" w/source - $79.95 - w/o source $49.95

**FULL SCREEN FORMS DISPLAY** from Computer Systems Consultants - TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
*F, S and CCF, U - $25.00, w/ Source - $50.00*

**SOLVE** from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No 'blind' debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!
*Levels I & II only - OS-9 $69.95*

## *DISK UTILITIES*

**OS-9 VDisk** from S.E. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.
*Level I OS-9 obj. $79.95; w/ Source $149.95*

**O-F** from S.E. Media -- Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK*DOS Format so it can be used normally by FLEX, SK*DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK*DOS Directory, Delete FLEX, SK*DOS Files, Copy both directions, etc. FLEX, SK*DOS users use the special disk just like any other FLEX, SK*DOS disk
*O - 6809/68000 $79.95*

**LSORT** from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.
*OS-9 $85.00*

**HIER** from S.E. Media - *HIER is a modern hierarchal storage system for users under FLEX, SK*DOS.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. **Using HIER a regular (any) FLEX, SK*DOS disk (8 - 5 - hard disk)** can have sub directories. By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. **Each directory looks to FLEX, SK*DOS like a regular file, except they have the extension '.DIR'.** A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK*DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!
*FLEX - SK*DOS $79.95*

**COPYMULT** from S.E. Media -- Copy LARGE Disks to several smaller disks. FLEX, SK*DOS utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.
*Completely documented Assembly Language Source files included. ALL 4 Programs (FLEX, SK*DOS, 8" or 5") $99.50*

**Availability Legends**
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC9 = Color Computer OS-9
CCF = Color Computer FLEX

VISA

**South East Media**
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

**** Shipping ****
Add 3% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

**COPYCAT** from Lucidata -- *Pascal NOT required.* Allows reading TSC Mini-FLEX, SK*DOS, SSB DOS68, and Digital Research CP/M Disks while operating under SK*DOS , FLEX1.O, FLEX 2.O, or FLEX 9.O with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.
*F, S and CCF 5" - $50.00 F, S 8" - $65.00*

**VIRTUAL TERMINAL** from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eight tasks on one terminal, under *VIRTUAL TERMINAL* and switch back and forth between tasks at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.
6809 O & CCO - obj. only - $49.95

**FLEX, SK*DOS DISK UTILITIES** from Computer Systems Consultants -- Eight (8) different Assembly Language (w/ Source Code) FLEX, SK*DOS Utilities for every FLEX, SK*DOS Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten XBASIC Programs including: A BASIC Resequencer with EXTRAs over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.
*ALL Utilities include Source (either BASIC or A.L. Source Code).*
*F, S and CCF - $50.00*
*BASIC Utilities ONLY for UniFLEX -- $30.00*

**MS-DOS-FLEX** Transfer Utilities to OS-9 For 68XXX and CoCo* OS-9 Systems Now READ - WRITE - DIR - DUMP - EXPLORE FLEX & MS-DOS Disk. These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks. *CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.
**CoCo Version: $69.95 68XXX Version $99.95*

## *MISCELLANEOUS*

**TABULA RASA SPREADSHEET** from Computer Systems Consultants -- TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.
*F, S and CCF, U - $50.00. w/ Source - $100.00*

**DYNACALC** -- Electronic Spread Sheet for the 6809 and 68000.
*F, S, OS-9 and SPECIAL CCF - $200.00, U - $395.00*
*OS-9 68K - $595.00*

**FULL SCREEN INVENTORY/MRP** from Computer Systems Consultants Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.
*F, S and CCF, U - $50.00. w/ Source - $100.00*

**FULL SCREEN MAILING LIST** from Computer Systems Consultants -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.
*F, S and CCF, U - $50.00, w/ Source - $100.00*

**DIET-TRAC** Forecaster from S.E. Media -- An XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.
*F, S - $59.95, U - $89.95*

## *GAMES*

**RAPIER** - 6809 Chess Program from S.E. Media -- Requires FLEX, SK*DOS and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. **This is one of the strongest CHESS programs running on any microcomputer**, *estimated USCF Rating 1600+ (better than most 'club' players at higher levels)*
*F, S and CCF - $79.95*

***NEW***

*MS-DOS/FLEX Transfer Utilities* For 68XXX and CoCo* OS-9 Systems. Now Read, Write, DIR, Dump and Explore FLEX & MS-DOS Disks. Supplied with a rich set of options to explore and transfer text type files from/to FLEX and MS-DOS disks. *CoCo OS-9 requires SDISK utilities & two floppy drives.
CCO $69.95 68XXX OS-9 $99.95

## *Macintosh Software at Discounted Prices*

***"Call for prices, it"ll be worth the savings."***

**NOTE: Changes**

1. Price increase for SCULPTOR, see advertising front of this catalog and other ad in this issue. Special price for 68 Micro Journal readers.
2. Lower price for BASICO9 TOOLS, see Utilities section.
3. New MS-DOS & FLEX to OS-9 Utilities, see above.

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC0 = Color Computer OS-9
CCF = Color Computer FLEX

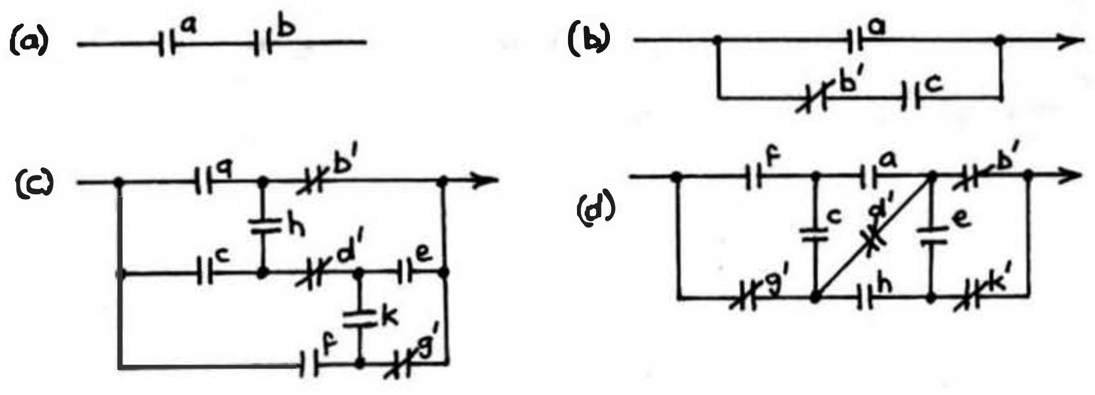
VISA

**South East Media**
*5900 Cassandra Smith Rd. - Hixson, Tn. 37343*

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.



Observe that networks (c) and (d) are not our normal ladder type. For instance, in (c), is the first loop "a + ch" OR maybe "c + ah"? We'll be designing networks like this, which cannot be described in ordinary Boolean algebra, a bit later on.

4. Design the following networks, using transfer contacts as economically as possible.

(a) L1 = abc + a'b'c'
(b) L1 = a(b + c') + de'    L2 = (a' + b'c)(d' + e)
(c) L1 = ab + c    L2 = ad + e
(d) L1 = a + bc    L2 = a + d + e
(e) L1 = a(b + c) + d + e'    L2 = a(b + c) + e + f'

5. Equalise the contact-load on a full relay-tree of four variables a, b, c and d, and draw the final tree.

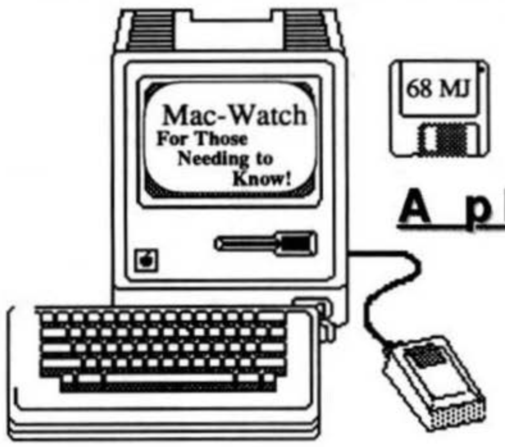
6. Minimise the number of transfer-contacts in the following partial trees

(a) a'b'c'd' a'b'cd' a'b'cd a'bc'd a'bcd ab'c'd' abcd' abcd
(b) a'bc'd' a'bcd' ab'c'd ab'cd abc'd' abc'd abcd' abcd
(c) abc'd' abc'd ab'c'd' a'bcd a'b'cd a'b'cd' ab'cd' abcd

It sometimes happens that it's impossible to obtain the APPARENT unused groupings by the sub-dividing process, and a group just HAS to be broken up. Exercise 6c is a good example of this condition! Don't forget that each map, or sub-map, must be divided into two EQUAL parts at each cutting operation. You CANNOT cut a 4-square submap into a 1-square and a 3-square, for instance. Anyway, have fun with this lot!!

... End of Mile 12, sitting at marker 'Mile 13'

+++



*The Macintosh™ Section*

**Reserved as**

**A place for your thoughts**

*And ours.......*

**Mac-Watch**

# A Review of SideKick 2.0

**Desktop Organizing Software From Borland, International**

By: James E. Law
1806 Rock Bluff Rd.
Hixson, TN, 37343

A number of software packages advertise that they can get us organized. Well, I for one, need to get organized so I was interested in trying the new version of Borland's SideKick.

This package has been around a long time, both for the Macintosh and IBM compatibles. SideKick contains a collection of desktop organization programs including telecon aids (e.g., dialing, phone books, and logs), calendars, a clock, a calculator, an enhanced notepad, a print spooler, and a simple specialized data base. In addition, the latest version (2.0) adds outlining and a spread-sheet.

SideKick applications, which comes on two 800k disks, can be used on any Macintosh. Several of the programs are provided in both application and desk accessory formats so that they can be used with current versions of MultiFinder.

With preliminaries aside, let's look at each of SideKick's major subprograms.

## First, Make an Outline

Remember in high school English classes when you were admonished to prepare an outline before writing your essay? If you had had your Macintosh and SideKick back then, the outline would have been no sweat. SideKick's outliner, called Outlook, is capable and relatively full featured. It may be all the outlines most users ever need.

Upon selecting Outlook from the Apple menu, you are given the option of opening an existing outline, starting a new outline, or resuming the last outline opened. Multiple outlines may be open at the same time.

Each entry in an outline begins with a small circle (i.e., a "bullet"). The bullet for a major heading is hollow if there are no lower tier entries under that heading. Otherwise, it is solid black. The bullet is followed by an icon that looks like a miniature page. This icon changes to indicate when there is text or graphics documents filed under the related heading. You may choose "Hide Icons" from the Outlook menu to get rid of the page icons.

To begin an outline you merely place the cursor to the right of the icon and begin typing. When you press return, a new bullet and page icon are entered and the cursor is automatically placed for your new entry. Any entry can be selected by clicking on its bullet. Selected entries can be moved to the right or left (to change their relative rank) by using the cursor keys or by dragging the entry with the mouse. Selected entries can also be deleted, copied, or cut.

Each entry on your outline may be associated with a text or graphics documents. By clicking on the page icon, this document can be opened, viewed, copied, cut, or (in the case of text) modified. For example, suppose that your outline was a list of actions needed to get ready for a presentation.

It might look like this:

- Preparation for sales presentation meeting
  - Prepare agenda
  - Prepare advertising material for review
    - Script for radio spot
    - Billboard layout
- Prepare overhead transparencies
  - Title transparency
  - Sales-by-district transparency
- Obtain meeting room

When you click the page icon for the entry entitled "script for radio spot," a text-only file would be opened with your script. Similarly, clicking the icon for "sales-by-district" might display a bar draft of sales statistics. You can see how an outline can be used as a powerful special-purpose data base including both text and graphics.

A specific document under an Outlook entry is either treated as text or graphics but not both. Also, text files are text only and do not include special formating information (i.e., multiple fonts and styles).

The Page Layout option from the Outlook menu allows you considerable flexibility in determining how your outline will look when it prints. For example, you can specify margins, line spacing, how entries will be numbered (e.g., 1, 1.1, 1.2; A, B, C), headings and footers. Your outline will print out per your specification with automatic page numbering. All text and graphics documents associated with your outline entries will be printed in the appropriate place in the outline. Unfortunately, all text is displayed in the same font, size, and style. A Text Settings option allows you to set these parameters but all text in the outline must be the same.

An outline by its very nature, is a big picture look at a subject. While there may be pages of detailed information backing up the outline, it is hidden. Outlook allows you to view as much or as little detail as you want to see. By double clicking any bullet, the next lower tier entries are hidden. Double clicking the bullet again will make them reappear. Triple clicking performs the same function for all entries under the selected entry regardless of their relative level. This feature looks smoothly and quickly to let you toggle between looking at the "forest" and the "trees."

Outlook works just as well with graphics as it does with text. You can place graphics under any heading and quickly view it full-sized by double clicking the related ballet. This feature could easily be used to construct an indexed library of clip art, charts, or other graphics.

## Let's Make a Plan

The latest version of SideKick also contains MacPlan, an effective little spreadsheet and graphics program in a desk accessory. MacPlan supports 50 rows by 20 columns. With up to 69 cells used for labels, a maximum of 931 cells are left for entering data. You may request MacPlan to format data as dollars, dollars and cents, percent, or as a number.

Use of this spreadsheet is straight forward. Any cell can be selected with the mouse or the cursor keys. Data may be entered directly or instead a formula may be added which can be used by MacPlan to calculate the value for a cell. For example, if a cell (say D7) were to contain "net profit," then you might indicate that C7 = C6 (gross sales) - C5 (expenses). Cell C6 (gross sales) might be calculated by MacPlan as the total of cells B1 through B12 (monthly sales). Then, if you revised your projection of July sales, the total sales and net profit figures would be automatically updated.

A wide range of formulas are provided for calculating the value of cells based on the value of other cells. In addition to basic operators (e.g., +, -, /,x, %), financial functions, depreciation functions, mathematical functions, logical functions, and calendar functions are included.

MacPlan allows you to create pie charts, line charts, and bar charts. Creation of a chart simply requires selecting "Graph" from the MacPlan menu, entering the title and legends, and indicating the type of chart you want. Bar charts may be plain, stacked, or grouped. You have no flexibility in determining fill patterns, fonts, on the size of the graph. Only 18 or so data points can be shown in pie charts and regular bar charts. With stacked or grouped bar charts up to 3 data entries can be made for each of the 18 points. For charts with many data points, part of the legend may be cut off and will not print. This feature has only limited usefulness

To print MacPlan spreadsheets and graphics you must copy the document then paste it into an application with printing capabilities.

## Just Keeping Time

I get tired of all those boring little digital clocks for the Macintosh. I know they are compact and easy to read, but they are not very interesting. I was glad to see that SideKick's timekeeper is a 1" x 2" analog clock with a second hand. The second hand continues to sweep even when another window is active. This clock seems to do what it was designed to do. It does not contain an alarm function.

**Monday, Monday**

At first glance, the SideKick calendar looks similar to the simple public domain calendars available for the Mac. A closer look, however, indicates that it is much more. Scroll bars allow any date between 1905 and 2030 to be viewed. Notes may be associated with any date. Dates for which notes are added appear circled. The notes window scrolls vertically to allow entry of more text than can be continued in the window. Pressing the return key automatically prints a dashed line to separate different notes.

A potentially useful feature of SideKick's calendar is the "Find" function. Unfortunately the usefulness is only *potential* and not real. First, it only searches the notes of the date currently selected date. It would have been useful to search all dates in a certain date range to see when the air conditioner filter was last changed. Also, I tried a number of searches on the notes of the currently selected date and none of them were completed successfully.

The Calendar Menu allows you to select the "Week at a Peek" option. This option allows you to view the notes for 7 days at once. The menu also allows either the regular calendar window or the "Week at a Peek" window with associated notes to be printed.

**When You Need To Do Some Figuring**

SideKick's calculator called "Calculator +" is a multifunction business calculator with "paper tape" output to the screen and printer. It allows entries from the keyboard or by clicking the keys on the calculator display.

Calculator + includes roots, logs, and trigometric functions as well as the usual mathematical functions. Also, it includes some very helpful financial function including:

- Calculate the growth of an initial invest ment
- Calculate the worth of a future return
- Calculate the worth of a future stream level of payment
- Calculate the level payment necessar to pay off a loan

Each of the above calculations can be made with only 2 or 3 entries.

Almost every Calculator + function can be operated from the keyboard which is much faster than using the mouse.

Calculator + is reasonably easy to read and use. It provides a new and powerful alternative to the Apple calculator.

**Area Code Lookup**

Area Code Lookup is a small desk accessory that gives information about any area code entered into a dialog box. For example, when "205" is entered the window reports:

| | |
|---|---|
| Locality: | Alabama |
| Region: | Birmingham |
| Time: | Central |

This may occasionally be useful in predicting whether a long distance contact may be a lunch or not.

**Write It Down**

"Notepad +" is a mini word processor that creates text files up to 28k which are compatible with MacWrite and Microsoft Word. The Notepad + menu allows you to open existing text files, save a file, print a file, or find characters, words, or phrases in a file.

The Notepad + window can be sized for convenient use with other windows on the screen. This DA is designed to be closely integrated with other SideKick functions. For example, Notepad + files can be sent by modem with MacTerm and printed in the background using Readiprinter.

Note that since Notepad + works only with text files, information related to page format , fonts, and styles is not included.

**Send It By Wire**

MacTerm is SideKick's Hayes-compatible 300, 1200, and 2400 band telecommunication desk accessory. It may be used to transmit and receive data while running other applications.

After selecting the "Configure MacTerm" icon, the usual adjustments may be made to match the software with your modem. Additionally, you may choose either the printer or modem port.

MacTerm remembers up to seven numbers that you frequently dial via modem and can automatically dial them for you.

**You Rang?**

One of the best known SideKick components is its telecon management tool also calles SideKick. (There is an desk accessory called MacDialer that performs many of the same functions). This feature can help you manage and organize every part of your telephone communications.

When SideKick is selected, the screen is filled with an attractive display of information and command buttons. It allows you to:

- Enter a phone number to be dialed by click ing on appropriate numbers in the key board display. Alternately you may select a name from the scrolling "PhoneBook" win dow and have SideKick look up the number and dial it for you. Dialing may be via modem or PhoneLink™. You may set the di aling speed as required.

- SideKick will time your call and calculate phone costs (for long distance calls) as well as consulting fees.

- You may make notes on each call by typing into the "PhoneNotes" window. These notes are saved as part of your phone log and may print out with other information about calls.

- You may go from this screen to the SideKick calendar by clicking the calendar button.

- You have the ability to easily add, delete, or change entries in thephone book. At your option, phone book entries may also appear in a menu of names. Entries in the Phone Book may be sorted in a variety of ways. A "miscellaneous notes" space allows you to enter information about each entry (e.g. birthday, shirt size, product). Also, you may have a number of different Phone-Books, one for home, one for business con tracts, etc. You can then switch between the phone books as needed.

As you make and receive phone calls, SideKick maintains a log of each call including your notes. This log can be viewed, sorted in a variety of ways, and printed.

The SideKick telephone organizer function does even more than I have summarized above. It's not hard to understand why it is one of the most popular such programs on the market.

**QuikSheets**

QuikSheets is a combination of four input forms that may be used to capture specific information. The forms included are:

- Alarms—Lets you enter the times of events you wish to be reminded of. When the alarm goes off a bell sounds and the Apple menu bar be gins to blink. It continues to blink until you acknowledge the alarm.

- Expenses—Provides for the accumulation of expenses with totals automatically calcu lated.

- Credit Cards—Provides for the recording of critical information about your credit cards.

- Things-to-do—Lets you enter things to do and check off accomplished items.

The QuikSheets desk accessory allows you to compact the cards by removing blank lines, completed tasks, and past alarms. QuikSheets may be printed.

**More, More, More**

In addition to the applications described above, SideKick contains a number of utilities. These utilities enhance the usefulness of the overall SideKick package. Included are:

- PrintManager—Allows you to obtain disired information from your Phone Book, CalendarBook, and Quik Sheet files and print it in a variety of formats.

- Converter—Allows you to copy phone book entries from Habadex, MacPhone, and various text files into your Phone Book.

- QuikEditor—Lets you design your own QuikSheets to keep track of all kinds of data.

**In Summary**

These seem to be the days of less for more. Much of todays software doesn't deliver on all its promises yet prices continue to escalate. SideKick is a delightful exception. It delivers! Each function that I used did its job in a simple and effective manner. Except for a minor complaint about the graphing function, I found SideKick to be an excellent value. When you can get a program with all these functions for $59, you should grab it!

**EOF**

# SOFTWARE USER NOTES

A Tutorial Series

By : Ronald W Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

*From Basic Assembler to HLL's*

## Assembler is Assembler is Assembler... or Not?

Last Saturday an envelope arrived in my mailbox. It was a copy of part of a leuer received by '68' Micro Journal from someone in Freedom PA (signature illegible in the copy). I quote one paragraph from a section headed "A few comments". "Another idea that would be nice. How about printing a simple program in XBASIC, BASIC09, C, PL9, Flex Assembler, OS9 Assembler and Pascal. ... The program should print text, loop with conditions and (do) simple arithmetic." My first reaction was a mental objection to the separation and distinction between OS9 Assembler and FLEX Assembler. After I thought about it a little, the objection went away. The programs may be quite different, and must be somewhat different because of operating system requirements.

Unfortunately, I don't have "all of the above," but I thought I'd take on the challenge with what I have. First, I do have TSC Extended BASIC, so I thought I'd dream up a program and then write it in as many languages as I could. I came up with a program that prints the heading "Multiplication Table," a blank line, and then the table for 9 times 0 through 12. (See listing 1). Each line of the table is to read out "3 X 9 = 27" etc. The BASIC program is all of 6 small lines, including the END statement which is not really necessary. It is certainly at least one of the shortest source listings in the group. Remember that XBASIC is an interpreter and that the interpreter is about 16K of program. Rather than just show program listings I though it might be informative to indicate how big the resulting programs are in terms of object code produced.

Lucidata Pascal is responsible for the next listing, (listing 2). just 7 lines of code. This Pascal is what is called a P-Code Pascal. It generates a code file that is sort of "semi-compiled." That is, the code is not machine code, but a sort of Pseudo code that may be interpreted by a simple interpreter. Because of this, the size of the object code is sort of indeterminate too. It is possible to compile and "bind" some of the runtime code to the compiled P-Code to produce a runnable object file, but a great deal of interpreter is included whether or not it is used, so the object code is larger than necessary. Please note that these two paragraphs are in no way a criticism of BASIC or P-Code Pascal, just an explanation of why object code size wouldn't mean much if reported here. Lucidata Pascal is a FLEX running 6809 compiler. I would hope that they might come up with a 68000 running version someday. Perhaps some of the following might serve to convince them to do so.

The programs are presented more or less in order of byte count. First let me say that unfortunately I have never run OS9 on the 6809 system. Perhaps some reader might fill us in on that version. I did write the program in 6809 Assembler for FLEX however. Listing 3 is that program. It uses three of the FLEX routines and this, of course is what would make the FLEX and OS9 versions different as you will see below. In order to help you understand the program, I've inserted the BASIC program lines as comments that precede the section of code that implements that line. In one case I' ve split a BASIC line into three parts. In order to keep the code closer. I've written the program in position independent code. The program is about one page and the byte count is 116.

I could not resist writing the program in 68000 Assembler, since I have both SK*DOS (remarkably like FLEX in operation and user available routines) and OS9 for the 68008 machine. The SK*DOS version is just about a direct translation from the FLEX 6809 version except that there are so many data registers available that I didn't have to define any variables in the program. All operations are carried out using data registers. I note that I couldn't use the PSTRNG routine of FLEX because it puts a CRLF BEFORE the text string that it prints out. SK*DOS has a routine with the identical function but it also has a PNSTRN function that doesn't put in a CRLF. I therefore didn't have to code a PSTRNG routine as I did in the FLEX version. Note that the PNSTRN SK*DOS routine requires A0 to be pointing at the start of the string and that the string must end with $04. OUT5D translates a 16 bit binary integer number to a decimal ASCII string and outputs it. The binary number must be in D4 and D5 is set to zero to inhibit leading spaces. The resulting program is only 82 bytes of object code, and is the champ in terms of brevity. I should mention that SK*DOS itself uses and can alter the registers A4-A7 and D4-D7. The other half, D0-D3 and A0-A3 are available to the user for his program. (See listing 4).

I had been toying with the idea of figuring out how to write assembler programs for OS9 use now that I had written some for SK*DOS, so I took it as a challenge one evening to write a program to print "Hi There" to my terminal. I had to read or at least skim about 100 pages of manual to find the necessary information, and between that, what seemingly was a sample program that wouldn't assemble properly, and a number of dumb errors on my part, it took about 4 hours to reach my goal. I was a bit frustrated but pleased. A day or two later I wrote the MULT program in OS9 68000 assembler. See listing 5.

The string of labels that follow the word "psect" are those necessary to generate the proper OS9 "header" for the file. The values for Edition, Typ_Lang and Atr_Rev are all defined in one of the files that is included by the "use defsfile". Psect stands for "Program Section", the place where non-alterable information is stored, that is, the program code and any constants may be put there. Vsect is the Variable section where alterable memory locations reside, that is locations where variables may be read or written.

OS9 is a multi-user system, and it has a nice feature. Two people can use the same copy of the program in memory at the same time because when they call it, they are each assigned a different variable area in memory. The register A6 contains the "base address" of the variables. When the computer switches tasks (users) it saves all the registers for the user whose task is temporarily suspended, and loads the registers with the values for the current task. A6 is different for each user of the same program. The variables are accessed by using the name as an offset from A6. (e.g. move.l d0,varname(a6)). This program has only one variable, called buffr. It is 14 bytes long and a reference to buffr(a6) will get the contents of the first byte, word or long depending on the reference. In the present case, A0 was set to point at the buffer by "lea buffr(A6),A0". Then A0 was manipulated to point to bytes in the buffer as it was loaded with characters.

One advantage of OS9 is that all the registers are available to the user program except A7 which is the stack pointer, and A6 which contains the pointer to the variable area. All the other registers may be used by the user program.

A disadvantage is that it places rather rigid constraints on the assembler programmer with regard to variables and accessing them. However, it is very good programming practice anyway, to separate the variable section from the program section. It can sometimes avoid accidentally writing over program instructions. OS9 guards data areas very carefully and any attempt to write to memory outside of your data area will cause an instant error message and the stopping of your program's execution.

The OS9 Assembler manual mentions that a file called sys.l should be used to include the system routine call names and values in the program. The manual wasn't specific about how to use the file, but it became apparent that it is a so called relocatable object file and that it is to be included at link time. The assembler produces relocatable object code, which must be linked by the linker before it can become an executable command file.

I decided to use the I$Write system routine which is similar to the PNSTRN call of SK*DOS. D0 must contain the "path number" for the output (1 to output to the STDOUT terminal). D1 must contain the length of the string, and A0 must point at the first byte of it. The system function F$Exit will report an error whose number is loaded in D1 before exiting the program. D1 must therefore be cleared before calling F$Exit. The syntax OS-9 F$Exit is an assembler convenience. The list of equates in the defsfile is scanned and the proper code placed in the file.

I wanted to use the math function that translates a long integer to an ASCII string, similar to OUT5H in SK*DOS, but I found that the routine includes all leading zeros, and the numbers output would have looked like "0000000018". I decided it would be simple to write my own conversion subroutine called "printi" which came out to 17 lines of code. What I have just said could be repeated for the "crlf" subroutine at the end of the program. All the register stuffing prior to the OS-9 call takes its toll in terms of program byte count, but remember that OS-9 doesn't tie up half of the registers for itself, and that it can do numerous tasks simultaneously. At any rate, the finished program was 174 bytes worth.

Now for the "native code" compiler versions. Long time readers might remember Whimsical, a compiler for the 6809 written by John Spray in New Zealand. Listing 6 shows the program, and again it is only a few lines. John has recently sent me a preliminary version of Whimsical to run under SK*DOS in 68000. The code required for this program is identical in both systems. The compiler produced 245 bytes of 68000 code and the 6809 version produced 328 bytes of 6809 code. Note that the 68000 version again was smaller than the 6809, just as happened in the FLEX / SK*DOS assembler cases. Though the 68000 instructions are more bytes, each one does more, and the same program can be done with less overall code. There are several ways to specify CR and LF in Whimsical, one is via ^M^J sequence. Another is CHAR $0D, CHAR $0A. Note that the Whimsical Syntax is pretty much in agreement with Pascal. John Spray has told me that this preliminary 68K version produces excess code and that a lot of optimization can be done to improve the object code size. I wonder how much better John can do?

Listing 7 is the PL9 6809 version and the OS9 PLuS 68000 version. The same input code will work for both compilers again. About the only disadvantage to PLuS / PL9 is that print statements can't have a strung out list of arguments. A different print procedure must be called for strings than for numbers. It is possible in this language to trim the library routines (IOSUBS in the present case) to remove any procedures that the program doesn't use. I did that for both versions. The 6809 version is 365 bytes and the 68000 version about 1300. The OS9 interface would seem to be responsible for the larger code in that version. Remember, in comparing, that the OS9 system has the advantage of multi users and multiple tasks running at the same time. When a computer has memory in the area of 750K or so, a few hundred bytes in a program are not an extravagance.

Listing 8 is one of two "C" versions that I did. This one would seem to win hands down for the shortest source code, but at least in my opinion, it comes in dead last in the compilers for readability. "C" has major advantages, the most important one being that it is very "portable." Most of the features of "C" are contained in what is called the "Standard Library." This library is written in "C" so that it moves from computer to computer without major rewrites. However, because it is written in "C" and not assembler, it is a little bulky. This version of the MULT program in Microware OS9 "C" for the 68000 produced 11,300 bytes of output code. The 6809 version (identical source code) in Windrush "C" by James McCosh produced 5246 bytes.

I decided that the "printf()" library function must be responsible for a lot of the code, so I did another version using the fputs() function (fILE putsTRING) for the strings, directing the output to the terminal. I wrote a simple printint() function (which is recursive) patterned after (read that "stolen from") the routine by Graham Trott of the same name in IOSUBS in PL9. Anyway the program compiled to 5140 bytes in the 68K version (less than half the size of the original), and to 4059 bytes in the 6809 version, about 25% smaller. See listing 9.

All of the programs produce essentially identical output with a few variations in number format and spacing. The heading was printed in keeping with the style of the language. Pascal and PL9 generally use upper case (no rules about it, just tradition, at least for me) so the headings are all caps. "C" programmers for some obscure (to me) reason prefer lower case as a convention for program statements and all caps for defined constants. The heading is all lower case in that program.

Generally, a program in assembler tends to be larger in source code and smaller in object code. It takes longer to debug an assembler program simply because there are more instructions to consider as a source of error. There are also more opportunities to make errors because more instructions must be used. The higher the level of the language, the smaller the source code. Some languages tend to be "self documenting," for example, the Whimsical, PL9 and Pascal versions. Though someone who has spent the past 5 years writing "C" programs every day would disagree, "C" programs tend to be more cryptic to the reader. Assembler programs may be done clearly if written in small "chunks." I see no reason to go as far as a main program that does nothing but call subroutines when numerous subroutines will only be used once, but it is a good idea to separate or group in-line code by function with a comment line or two between the various portions. Of course if a particular function is needed more than two or three times in a program a subroutine is called for simply to reduce the code produced. A good side effect is to improve the readability of the code.

Now a word about recursive procedures. Note that the printint() in the "C" program references itself. Most of the languages that are considered "procedural" in nature (they break a program into functions or procedures or subroutines) allow two kinds of variables. There are "Global" variables and there are "Local" ones. Global variables are "declared" at the beginning of the program and they exist as long as the program is running. Local variables are associated with particular procedures or subroutines. When the procedure is running, space for local variables is allocated (usually on the stack). When the procedure ends, the space is deallocated. Local variables won't retain information once the procedure is finished and control returned to the main program ("c" has a special case called a "static" variable that may be defined within a function, that remains between calls to that function). At any rate, this method of allocating variable space for a procedure when it is "called" leads to an interesting possibility. In the present case, look at the printint() function in listing 9. The function is passed the number to be converted from binary to ASCII characters for printing. The function has a local variable called "digit." digit = number % 10 + ox30; This statement (% is the "mod" or modulus function. Number Mod 10 results in the REMAINDER resulting from dividing number by 10, which is the lowest order digit of the number to be printed. We add $30 to it and the result is the ASCII code for that digit. Then we divide number by 10 (integer divide) which essentially throws away the remainder that we just turned into a character. That character is placed in the local variable "digit" remember. .p Now comes the tricky part. If the number is not zero, (number, which was passed to the function is also a local variable on the stack), the function calls itself, creating new local variables number and digit in a new place on the stack. The value of number passed to this second call to printint() is not the starting value but the value resulting from the divide by 10 in the first call to printint(). The new call to the procedure again does the mod function, placing the next higher order digit in the new local variable "digit." The key to what is going on is to remember that the first call to the function printint has not yet finished executing. Its execution was interrupted by the second call from within the function itself. The process continues, each call to printint() getting as far as the next call, number getting smaller each time by the successive divides by 10, until number becomes zero. At that point the last call to printint() goes on and executes the last line, putc (digit, stdout);. As soon as that happens, the last call to printint() is finished and it returns to the previous call which has the next digit in its local variable "digit" It finishes by printing the character and returns to the previous call until the first call is reached when it prints the last character and returns to the main program.

Note that the OS9 68K assembler version of this routine does essentially the same thing but it is not recursive. It puts the digit that it has calculated at the end of a buffer and decrements a pointer, then continues in a loop to calculate earlier digits until number is reduced to zero. Then it loads the digit count and calls the print string function of OS9. One could make the assembler version recursive simply by pushing the currently calculated digit onto the system stack move.b dn,-(a7). Because of the way a character must be output with I$Write, when the last call to the procedure is completed, one would have to pull the characters off the stack and stuff them in a buffer to have them assembled for printing anyway, and that would just be more code.

If all this is thoroughly confusing remember that "C" has the built-in function to print a number but we were trying to be a little more efficient and use our own. Pascal, Whimsical, and PL/9 all have such functions as well.

I hope this discussion has given you a little glimpse of programming from several points of view. I point out that several of the programs are identical in 6809 and 68000 versions when written in one of the higher level languages. The assembler code is quite different for the two processors, and even with one processor, the code is significantly different depending on the operating system. Assembler code is very efficient, runs fast, uses less bytes of code, but it is also VERY specific to the processor and operating system. The higher level languages shield the user from the processor but not generally quite completely from the operating system. The format and allowable size of filenames, for example are usually a function of the operating system and not of the language. Some of the languages allow you to do a "system call," which of course is operating system dependent. For example XBASIC will let you EXEC,"ITYSET PS=N" to turn the pause function off before outputting to a printer. Microware "C" allows system calls to OS9 to do things like rename or delete a file or to set up terminal parameters. The call system "nopause" does the same thing as the EXEC example for XBASIC above.

**Listing 1**

```
  10 PRINT "MULTIPLICATION TABLE"
  12 PRINT
  15 FOR N=0 TO 12
  20 PRINT N;"X 9 =";
  30 PRINT N*9
  40 NEXT N
```

**Listing 2**

```
PROGRAM MULT(OUTPUT);

   VAR NUM, RESULT:INTEGER;

BEGIN
   WRITELN('MULTIPLICATION TABLE');
   WRITELN;
   FOR NUM:= 0 TO 12 DO WRITELN(NUM,' X 9
=',NUM*9);
END.
```

**Listing 3**

```
 NAM MULT
 TTL PROG TO PRINT MULTIPLICATION TABLE
*
* FLEX EQUATES
*
OUTDEC EQU $CD39
PCRLF  EQU $CD24
PUTCHR EQU $CD18
WARMS  EQU $CD03
 ORG 0
START BRA BEGIN
MSG FCC /MULTIPLICATION TABLE/,$04
MSG1 FCC / X 9 = /,$04
NUM RMB 2 SAVE TWO BYTES FOR N
RESULT RMB 2
* 10 PRINT "MULTIPLICATION TABLE"
BEGIN LEAX MSG,PCR
 BSR PSTRNG
 JSR PCRLF
* 20 PRINT
 JSR PCRLF
* 30 FOR N=0 TO 12
 CLR NUM,PCR
 CLR NUM+1,PCR
LOOP LDD NUM,PCR
 CMPB #12
 BGT ENDLOP
*40 PRINT N;
 LEAX NUM,PCR
 CLRB
 JSR OUTDEC
* 40 CONTINUED PRINT"X 9 = ";
 LEAX MSG1,PCR
 BSR PSTRNG
* 40 CONTINUED PRINT N*9
 LDD NUM,PCR
 LDA #9
 MUL
 STD RESULT,PCR
 CLRB
 LEAX RESULT,PCR
 JSR OUTDEC
 JSR PCRLF
```

```
*50 NEXT N
 LDD NUM,PCR
 ADDD #1
 STD NUM,PCR
 BRA LOOP
ENDLOP JMP WARMS
*
* PSTRNG SUBROUTINE
*
PSTRNG LDA ,X+
 CMPA #$04
 BEQ DONEP
 JSR PUTCHR
 BRA PSTRNG
DONEP RTS
*
 END START
```

**Listing 4**

```
 NAM MULT
 TTL MULTIPLY
*
* SK*DOS EQUATES
*
PNSTRN EQU $A036  A4 IS POINTER
OUT5D EQU $A038   D4 WORD D5=0 TO SUPPRESS
SPACES
PCRLF EQU $A034
WARMST EQU $A01E
*
 ORG 0
START BRA BEGIN
*
MSG DC.B "MULTIPLICATION TABLE",$04
MSG1 DC.B " X 9 = ",$04
* NUM WILL BE IN D0
* RESULT WILL USE D4
*
* 10 PRINT "MULTIPLICATION TABLE"
BEGIN LEA MSG(PC),A4
 DC PNSTRN
 DC PCRLF
* 20 PRINT
 DC PCRLF
* 30 FOR N=0 TO 12
 CLR D0  N
LOOP CMP #12,D0
 BGT.S ENDLOP
* 40 PRINT N;
 MOVE D0,D4
 CLR D5
 DC OUT5D
* 40 PRINT "X 9 ="
 LEA MSG1(PC),A4
 DC PNSTRN
* 40 PRINT N*9
 MOVE #9,D4
 MULU D0,D4
 CLR D5
 DC OUT5D
 DC PCRLF
* 50 NEXT N
 ADD #1,D0
 BRA.S LOOP
ENDLOP DC WARMST
 END START
```

**Listing 5**

```
 nam mult
 ttl os9 mult prog
 use defsfile

 psect mult,Typ_Lang,Attr_Rev,Edition,0,Mult

Edition equ 1
Typ_Lang equ (Prgrm<<8)+Objct
Attr_Rev equ (ReEnt<<8)+0

StdOut equ 1

message dc.b $0d,$0a,"Multiplication
table",$0a,$0a,$0d
mesglen equ *-message
mesgl   dc.b " X 9 = "
msgllen equ *-mesgl

 vsect
buffr   ds.b 14
 ends

* 10 print "Multiplication Table"
* 20 print
Mult    moveq.l #1,d0  path number
        move.l #mesglen,d1 number of bytes
        lea message(pc),a0  buffer pointer
        os9 I$Write
* 30 for n=0 to 12
        clr d2
Loop    cmp #12,d2
        bgt.s Endlop
* 40 print n," X 9 = ",n*9
        move.l d2,d1 pass number in d1
        bsr.s printi print an integer
        moveq.l #1,d0
        move.l #msgllen,d1
        lea mesgl(pc),a0
        os9 I$Write
        move.l d2,d0
        move.l #9,d1
        mulu.w d0,d1 product in d1
        bsr.s printi
        bsr.s Crlf
        addq.w #1,d2
        bra.s Loop
Endlop  move.l #0,d1
        os9 F$Exit

* Enter with number in d1 convert and put in
buffr
* and print via I$WritLn
*
printi  lea buffr(a6),a0
        adda.l #12,a0
        clr.l d3 counter for bytes in string
```

```
ploop   move.l #10,d0
        divu d0,d1 mod in upper word of D1
        swap.w d1
        add.b #$30,d1
        move.b d1,-(a0)
        clr.w d1 clear remainder
        addq #1,d3
        swap.w d1
        tst.w d1
        bne.s ploop
        move.l #1,d0 path
        move.l d3,d1 length of string
        os9 I$Write
        rts
* crlf routine
Crlf    move.w #$0d0a,buffr(a6)
        lea buffr(a6),a0
        moveq.l #1,d0
        moveq.l #2,d1
        os9 I$Write
        rts

        ends
```

**Listing 6**

```
% mult program in whimsical 6809

BEGIN
  INTEGER NUM;

  WRITE "MULTIPLICATION TABLE^M^J^J";
  NUM :=0;
  WHILE NUM <=12 DO BEGIN
    WRITE NUM," X 9 = ",NUM*9,"^M^J";
    NUM :=NUM+1;
  END;
END.
```

**Listing 7**

```
/* PROGRAM MULT IN PL9 */

ORIGIN = 0;
STACK = $BE00

GLOBAL INTEGER NUM;

INCLUDE IOSUBS.LIB.1;

PROCEDURE MAIN;
   PRINT "MULTIPLICATION TABLE";
   CRLF;
   CRLF;
   NUM=0;
   WHILE NUM < 13 BEGIN
      PRINTINT(NUM);
      PRINT " X 9 = ";
      PRINTINT(NUM*9);
      CRLF;
      NUM = NUM+1;
   END;
```

**Listing 8**

```
/* mult table program in C */


#include <stdio.h>

main()
{
    int num;

    printf("multiplication table\n\n");
    for (num=0; num <=12; num++) printf("%2d
X 9 = %3d\n",num,num*9);
}
```

**Listing 9**

```
/* mult table program in C 6809 */

#include <stdio.h>

main()
{
   int num;

   fputs("multiplication table\n\n",stdout);
   for (num=0; num <=12; num++)
   {
      printint(num);
      fputs(" X 9 = ",stdout);
      printint(num*9);
      putc('\n',stdout);
   }
}

printint(number)
   int number;
{
   char digit;
   digit = number %10 + '0';
   number = number/10;
   if(number !=0) printint(number); /* recur-
sive */
   putc(digit,stdout);
}

EOF
```

*FOR THOSE WHO NEED TO KNOW* **68 MICRO JOURNAL™**

# FORTH

A Tutorial Series

By: R. D. Lurie
9 Linda Street
Leominister, MA 01453

## THE DRAG STRIP

I know that some of you must think that I have a fixation on execution speed, but I have found the topic to be fascinating. There are very practical reasons to be concerned about execution speed in FORTH, since it is often chosen as the embedded language for controllers because of its high speed. Furthermore, we are all interested in having our applications run as fast as possible, because we are irritated beyond all reason by having to wait on a computer. Therefore, I have continued my study of ways to get from here to there as fast as possible, hence the subtitle "the drag strip."

In this installment, I want to discuss ways of speeding up multiplication operations. Examination of Figure 8 in my last column shows that the word 2* executes about 6.6 times faster than the logically equivalent pair of words, 2 * . This difference cries out for investigation!

This is typical of a situation where a generalized technique is compared to a specialized case. In other words, 2* consists only of load, add, and store operations, so it must be faster than the much more complex, and thereby general, * operation.

### High Level FORTH

This set of experiments had two goals: (1) to speed up simple multiplication operations, and (2) to do it in high-level FORTH. Goal #1 is obvious, but goal #2 was chosen to maintain portability, even from processor to processor. In other words, I wanted to adhere to the principle of KISS (Keep it simple, stupid!).

Any multiplier greater than 2 can be generated by a series of doublings and algebraic additions. For example, consider 13 as a multiplier, as in Figure 1.

If you follow through the FORTH logic, you will see that the FORTH notation is identical to the logic of the last line of the algebraic notation. Any other multiplier can be generated in the same way. The FORTH screens 20 and 22-24 show this.

Screen 21 shows the actual timing loop used for testing 10* . The multiplicand, 23, was defined as a constant by the line 23 CONSTANT 23 . There is no particular significance to using 23 as opposed to some other value; I just happened to choose it at random. The testing was done on my CoCo3 at 1.89 MHz. The external timer was controlled by the cassette recorder relay through the words TIME_ON and TIME_OFF . The data do show a bit more scatter than I would like, but the total time must be accurate to within +/- 0.5 second. I decided not to pursue this scatter any further at this time, since the data are certainly good enough to prove the point I am trying to make. The tests and results are summarized in Figure 2.

Notice that the definition of 15* differs from the others in that it contains a subtraction, rather than an addition. I did this simply in order to shorten the definition. The definition would have needed two more terms in order to contain an addition, instead of a subtraction, and this would have also lengthened the execution time.

The last line of Figure 2 contains a measurement of the time required to multiply by 15 as one would normally do it. Notice that this operation was about 18 microseconds shorter per pass through the loop than the loop containing 15* . This simply proves that one can outsmart himself if he is not careful! If the definition gets too long, about 7 terms, then it is quicker to run the conventional path. However, notice that this definition of 16* is about 42 microseconds faster than 15 * , so it obviously pays to look for special cases.

### FORTH plus assembly language

Now let us consider the case where speed is most important, and we don't care about transportability. To do this, we should write the critical part of the definition in assembly language. This should give us the minimum execution time, short of writing the whole application in assembly.

This also demonstrates a great virtue of FORTH, in that it is so very easy to incorporate assembly language definitions into a program. Sure, you can do the same kind of thing with C or Pascal, but not nearly so easily. The important points to observe here are that the assembly language definition is called just like any other definition and data are transferred on the Data Stack just the same way as with high-level definitions.

Since I don't have access to a 68000 system, I will have to stick to the 6809. Fortunately, FF9 has an assembler, so that part of the job will be easy. I am not a whiz at writing math routines in assembly language, so my efforts may not be the most efficient, but they do work. If anyone has faster routines, please share them with us.

The definitions I wrote are shown in screens 26-38. With the exception of the definition for 15* , all of the definitions are direct transliterations from the high-level definitions. I simply took Wilson Federici's FF9 definitions for DUP , + , and 2* and incorporated the important parts into each new definition. The definition of 15* now follows the pattern of the rest of the definitions as ( ( 7 * 2 ) + 1 ).

The test results are summarized in Figure 3.

These definitions average about 2.4 times faster than the corresponding high- level definition, even the very long 15* . I don't think that you could make these operations run much faster, within the constraints of calling data from a stack and calling the multiplier as a virtual subroutine. I think that in-line code would be the only way to go any faster!

**Caveats**

Caution, this is signed integer math, limited to 16 bits. In other words, the product of any multiplication can never fall outside the range of +32767 through -32768. Any product exceeding these limits will, at best, be truncated on the high end, and, at worst, bear no resemblance to any part of the real answer. Remember that FORTH has a different set of definitions for use with 32- bit numbers.

Also, these definitions can only be used where the multiplier is a constant. If you must multiply two variables, you must use the conventional * operation. High-level FORTH versus assembly language and others

The factor of 2.4 penalty paid for running in high-level FORTH is hardly any penalty at all!. What other high-level language can do so well? Of course, now that I have the assembly definitions written, I will probably use them most of the time, instead of the high-level versions, but I will probably rarely, if ever, notice the difference. In the cases shown here, most of the added time comes from the overhead encountered in "calling subroutines." Since C and Pascal function, mainly, by also calling subroutines, the compiled form of either one could hardly be expected to be faster than well written FORTH. For what it is worth, I think that it is harder to write good, tight programs in C or Pascal than it is in FORTH, so that is how I account for the general pronouncement that FORTH is usually faster than either one. By the way, I have seen threaded FORTRAN execute faster than conventionally compiled FORTRAN, both on a PDP-11. Besides, FORTH is more fun!

**MORE ON FLEX AND THE COCO3**

I forgot to mention in my last column that you must use the WIDTH32 screen option when you are loading FLEX into your CoCo3, if you have the same version as I use. If you try to load FLEX while in WIDTH40 or in WIDTH80, the system will lock up, and you will have to reboot. It took me a while to work this one out, so I can't imagine why I forgot to stress it last time. Once loaded, you can select the V51.CMD display, just as if you were using a CoCo1 or 2, but you cannot use the "hardware" 80 or 40 column screens. The problem is that FLEX clobbers some of the pointers and drivers when it loads; you must have a new driver if you want an 80 column screen. No, I have not written one!

I also said last time that I was having trouble getting my printer to work. Well, that problem has gone away, and I have no idea why!?! In any case, the printer works fine, now.

**FORTH WITH OS9, LEVEL I**

I have now had a chance to spend some time on the DELPHI FORTH. It runs quite well on OS9, level I, version 2.0. This FORTH is called FORTH09 by the author, Dennis M. Weldy. He says that it is a preliminary version, which I hope means that he will soon publish a more complete version.

Once you figure out some of the peculiarities forced by OS9, this is a very "user friendly" FORTH. It has no editor directly associated with it. Developing programs is no real problem, since you can enter trial definitions directly from the keyboard. Once you have a definition working properly, you use the normal OS9 editor, or any other text editor, to prepare an ASCII file, such as the one shown in Figure 4.

As you would expect, this simple program prints the message and the numbers 0 through 99. The indentation is not required, but makes the definition easier to read. Since this is a text file, there are no screen numbers to be concerned with, which some people consider an advantage (I think that the jury is still hung on that question). The program is called by the following command:

```
FORTH09 <TEST1
```

Notice that this is a standard OS9 command line with input redirected to the disk file named TEST1. That is why the last line of the file consists of the name of the definition to be executed; I don't know of another way to get input into the FORTH09, once you have redirected the input. In any case, this works nicely, and I have no complaints.

Since this is a "preliminary" version, and OS9 frowns on the practice, there are no words for direct interaction with the machine. Table 1 lists the words which are available, and you can sure do a lot with what is here. Certainly, if you are a good C programmer, you can add more words, since the source code is available.

The basis for FORTH09 was the first edition of Brodie's "Starting FORTH." Therefore, a few of the words are from Polyforth or fig-FORTH, and not FORTH- 83, but that is a minor quibble. The bigger problem is the lack of fetch and store definitions.

All in all, I like FORTH09 for what it is, but more for what it can become!

**EOF**

```
Algebraic notation:
     13 = ( 2 * 6 ) + 1
     13 = ( 2 * ( 2 * 3 ) ) + 1
     13 = ( 2 * ( 2 * ( ( 2 * 1 ) + 1 ) ) ) + 1

FORTH notation:
     13* =  DUP  2*  DUP  2*  +  2*  +
```

*Figure 1. Derivation of 13* .*

| multiplier | | measured time (sec.) | adjusted time (micro) |
|---|---|---|---|
| 3* | ( DUP 2* + ) | 147.0 | 85.5 |
| 4* | ( 2* 2* ) | 131.3 | 69.8 |
| 5* | ( DUP 2* 2* + ) | 167.1 | 105.6 |
| 6* | ( 2* DUP 2* + ) | 167.0 | 105.5 |
| 7* | ( DUP 2* DUP 2* + + ) | 202.8 | 141.3 |
| 8* | ( 2* 2* 2* ) | 151.5 | 90.0 |
| 9* | ( DUP 2* 2* 2* + ) | 188.0 | 126.5 |
| 10* | ( DUP 2* 2* + 2* ) | 187.1 | 125.6 |
| 11* | ( DUP DUP 2* 2* + 2* + ) | 223.0 | 161.5 |
| 12* | ( 2* DUP 2* + 2* ) | 187.2 | 125.7 |
| 13* | ( DUP 2* DUP 2* + 2* + ) | 222.9 | 161.4 |
| 14* | ( DUP 2* DUP 2* + + 2* ) | 222.9 | 161.4 |
| 15* | ( DUP 2* 2* 2* 2* SWAP - ) | 230.8 | 169.3 |
| 16* | ( 2* 2* 2* 2* ) | 171.5 | 110.0 |
| 15 * | | 213.0 | 151.5 |

*Figure 2. Summary of the timing test results for high-level definitions.*

| multiplier | measured time (sec.) | adjusted time (micro) |
|---|---|---|
| 3* | 97.2 | 35.7 |
| 4* | 89.3 | 27.8 |
| 5* | 105.6 | 44.1 |
| 6* | 105.6 | 44.1 |
| 7* | 122.4 | 60.9 |
| 8* | 97.8 | 36.3 |
| 9* | 114.6 | 53.1 |
| 10* | 114.6 | 53.1 |
| 11* | 132.1 | 70.6 |
| 12* | 114.5 | 53.0 |
| 13* | 132.2 | 70.6 |
| 14* | 131.3 | 69.8 |
| 15* | 148.0 | 86.5 |
| 16* | 106.7 | 45.2 |

*Figure 3. Summary of the timing test results for assembly language definitions.*

```
    : TEST1     ( - )
                CR  ." This is a test of FORTH09 by D M
Weldy."  CR
                100  0  DO
                        I  .
                LOOP ;

        TEST1
```

Figure 4. Sample FORTH09 input.

| | | | | | |
|---|---|---|---|---|---|
| + | - | * | / | MOD | |
| /MOD | SWAP | DUP | OVER | ROT | DROP |
| CR | SPACES | SPACE | EMIT | 2SWAP | 2DUP |
| 2OVER | 2DROP | ." | = | < | > |
| 0= | 0< | 0> | NOT | AND | OR |
| ?DUP | ABORT" | ?STACK | IF | ELSE | THEN |
| 1+ | 1- | 2+ | 2- | 2* | 2/ |
| ABS | NEGATE | MIN | MAX | >R | >R |
| I | I' | J | */ | */MOD | DO |
| LOOP | +LOOP | LEAVE | BEGIN | UNTIL | WHILE |
| REPEAT | U.R | PAGE | QUIT | | |

Table 1. The 64 primitive definitions within FORTH09.

```
SCR #20
   \ Math experiments-MULTIPLICATION                    \ RDL 02/11/88

   23 CONSTANT 23

   : 3* ( n - )                                          \ RDL 02/11/88
      DUP  2*  + ;

   : 3X ( n - )                                          \ RDL 02/11/88
      3  * ;

SCR #21
   ( Timing loop                                          \ RDL 02/10/88)

   : TEST    ( - )                                       \ RDL 02/11/88
      TIME_ON
      20  0  DO
        $0000  0  DO
   23 11*
          SP!
        LOOP
      LOOP
      TIME_OFF
      7  EMIT ;

SCR #22
   \ Math experiments-MULTIPLICATION                    \ RDL 02/11/88

   23 CONSTANT 23

   : 4* ( n - )                                          \ RDL 02/11/88
      2*  2* ;

   : 5* ( n - )                                          \ RDL 02/11/88
      DUP  2*  2*  + ;

   : 6* ( n - )                                          \ RDL 02/11/88
      2*  DUP  2*  + ;

   : 7* ( n - )                                          \ RDL 02/11/88
      DUP  2*  DUP  2*  +  + ;

SCR #23
   \ Math experiments-MULTIPLICATION                    \ RDL 02/11/88

   : 8* ( n - )                                          \ RDL 02/11/88
      2*  2*  2* ;

   : 9* ( n - )                                          \ RDL 02/11/88
      DUP  2*  2*  2*  + ;

   : 10*      ( n - )                                    \ RDL 02/11/88
      DUP  2*  2*  +  2* ;

   : 11*      ( n - )                                    \ RDL 02/11/88
      DUP  DUP  2*  2*  +  2*  + ;

   : 12*      ( n - )                                    \ RDL 02/11/88
      2*  DUP  2*  +  2* ;

SCR #24
   \ Math experiments-MULTIPLICATION                    \ RDL 02/11/88

   : 13*      ( n - )                                    \ RDL 02/11/88
      DUP  2*  DUP  2*  +  2*  + ;

   : 14*      ( n - )                                    \ RDL 02/11/88
      DUP  2*  DUP  2*  +  +  2* ;

   : 15*      ( n - )                                    \ RDL 02/11/88
      DUP  2*  2*  2*  2*  SWAP  - ;

   : 16*      ( n - )                                    \ RDL 02/11/88
      2*  2*  2*  2* ;

SCR #25
   \ Math experiments-MULTIPLICATION                    \ RDL 02/11/88

   \ DUP                                                 \ WMF 02/11/88
   \     0 ,U LDD  _D PSHU

   \ +                                                   \ WMF 02/11/88
   \    _D PULU  0 ,U  ADDD  0 ,U STD

   \ 2*                                                  \ MMF 02/11/88
   \     0 ,U LDD  0 ,U ADDD  0 ,U STD

SCR #26
   \ Math experiments-MULTIPLICATION                    \ RDL 02/11/88

   CODE  3*       ( n1 - n2 )                            \ RDL 02/11/88
       0 ,U LDD  _D PSHU                     \ DUP
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       _D PULU  0 ,U  ADDD  0 ,U STD         \ +
       NEXT,
   END-CODE

   CODE  4*       ( n1 - n2 )                            \ RDL 02/11/88
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       NEXT,
   END-CODE

SCR #27
   \ Math experiments-MULTIPLICATION                    \ RDL 02/11/88

   CODE  5*       ( n1 - n2 )                            \ RDL 02/11/88
       0 ,U LDD  _D PSHU                     \ DUP
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       _D PULU  0 ,U  ADDD  0 ,U STD         \ +
       NEXT,
   END-CODE

SCR #28
   \ Math experiments-MULTIPLICATION                    \ RDL 02/11/88

   CODE  6*       ( n1 - n2 )                            \ RDL 02/11/88
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       0 ,U LDD  _D PSHU                     \ DUP
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       _D PULU  0 ,U  ADDD  0 ,U STD         \ +
       NEXT,
   END-CODE

SCR #29
   \ Math experiments-MULTIPLICATION                    \ RDL 02/12/88

   CODE  7*       ( n1 - n2 )                            \ RDL 02/12/88
       0 ,U LDD  _D PSHU                     \ DUP
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       0 ,U LDD  _D PSHU                     \ DUP
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       _D PULU  0 ,U  ADDD  0 ,U STD         \ +
       _D PULU  0 ,U  ADDD  0 ,U STD         \ +
       NEXT,
   END-CODE

SCR #30
   \ Math experiments-MULTIPLICATION                    \ RDL 02/12/88

   CODE  8*       ( n1 - n2 )                            \ RDL 02/12/88
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       NEXT,
   END-CODE

SCR #31
   \ Math experiments-MULTIPLICATION                    \ RDL 02/12/88

   CODE  9*       ( n1 - n2 )                            \ RDL 02/12/88
       0 ,U LDD  _D PSHU                     \ DUP
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       0 ,U LDD  0 ,U ADDD  0 ,U STD         \ 2*
       _D PULU  0 ,U  ADDD  0 ,U STD         \ +
       NEXT,
   END-CODE

SCR #32
   \ Math experiments-MULTIPLICATION                    \ RDL 02/12/88
```

```
  CODE  10*      ( n1 -- n2 )                        \ RDL 02/12/88
     0 ,U LDD   _D PSHU                      \ DUP
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
    _D PULU   0 ,U  ADDD   0 ,U STD          \ +
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     NEXT,
  END-CODE

SCR #33
   \ Math experiments--MULTIPLICATION                \ RDL 02/12/88

  CODE  11*      ( n1 -- n2 )                        \ RDL 02/12/88
     0 ,U LDD   _D PSHU                      \ DUP
     0 ,U LDD   _D PSHU                      \ DUP
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
    _D PULU   0 ,U  ADDD   0 ,U STD          \ +
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
    _D PULU   0 ,U  ADDD   0 ,U STD          \ +
     NEXT,
  END-CODE

SCR #34
   \ Math experiments--MULTIPLICATION                \ RDL 02/12/88

  CODE  12*      ( n1 -- n2 )                        \ RDL 02/12/88
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     0 ,U LDD   _D PSHU                      \ DUP
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
    _D PULU   0 ,U  ADDD   0 ,U STD          \ +
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     NEXT,
  END-CODE

SCR #35
   \ Math experiments--MULTIPLICATION                \ RDL 02/12/88

  CODE  13*      ( n1 -- n2 )                        \ RDL 02/12/88
     0 ,U LDD   _D PSHU                      \ DUP
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     0 ,U LDD   _D PSHU                      \ DUP
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
    _D PULU   0 ,U  ADDD   0 ,U STD          \ +
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
    _D PULU   0 ,U  ADDD   0 ,U STD          \ +
     NEXT,
  END-CODE

SCR #36
   \ Math experiments--MULTIPLICATION                \ RDL 02/12/88

  CODE  14*      ( n1 -- n2 )                        \ RDL 02/12/88
     0 ,U LDD   _D PSHU                      \ DUP
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     0 ,U LDD   _D PSHU                      \ DUP
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
    _D PULU   0 ,U  ADDD   0 ,U STD          \ +
    _D PULU   0 ,U  ADDD   0 ,U STD          \ +
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     NEXT,
  END-CODE

SCR #37
   \ Math experiments--MULTIPLICATION                \ RDL 02/12/88

  CODE  15*      ( n1 -- n2 )                        \ RDL 02/12/88
     0 ,U LDD   _D PSHU                      \ DUP
     0 ,U LDD   _D PSHU                      \ DUP
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     0 ,U LDD   _D PSHU                      \ DUP
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
    _D PULU   0 ,U  ADDD   0 ,U STD          \ +
    _D PULU   0 ,U  ADDD   0 ,U STD          \ +
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
    _D PULU   0 ,U  ADDD   0 ,U STD          \ +
     NEXT,
  END-CODE

SCR #38
   \ Math experiments--MULTIPLICATION                \ RDL 02/12/88

  CODE  16*      ( n1 -- n2 )                        \ RDL 02/12/88
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     0 ,U LDD   0 ,U ADDD   0 ,U STD         \ 2*
     NEXT,
  END-CODE
```

+++

*FOR THOSE WHO NEED TO KNOW*

**68 MICRO JOURNAL™**

## BETTER WINCHESTER DRIVERS

By: Samuel I. Green, Ph.D.

In December 1986, the 68 Micro Journal published my paper called Simple Winchester which uses the Western Digital WD1000-05 controller. My article included modifications to drivers previously published by Graves and Zeff and was issued as Reader Service Disk-30. Since the article was published, I have changed drivers. I now use the Winchester drivers provided by Leo Taylor. In response to a letter in the 68 Micro Journal, I sent Leo Taylor six blank diskettes and he filled them with a treasure trove of utilities and documentation which I still haven't finished perusing. Included was a set of Winchester drivers far superior to mine. I changed over in 1986 and have never looked back.

By comparison, my drivers allocate one volume for each of two heads. WINSYS uses all available space on multiple drives and allows assignment to as many as 26 total volumes. My drivers never really worked right at 2 MHz even with I/O slowdown as required by the controller. WINSYS works fine at 2 MHz.

I finally contacted Leo Taylor about submitting these drivers to the 68 Micro Journal. He said I can submit them to you if he is credited as the source. I herewith submit Leo Taylor's Winchester drivers and a utility called SQUEEZE which is needed to modify FLEX for booting directly from the Winchester. I urge you to discontinue distributing my Reader Service Disk-30 and replace it with the enclosed diskette. *(editors's note: Done)*

WINSYS is a group of seven files to support Winchester drives connected to a WD1002-05 which has error detection and correction or to a WD1000-05 which does not. No change is required in the hardware interface described in my article. WINSYS includes the formatter, drivers, assigner, booter, parker, and documentation. Multiple Winchester drives are supported and a maximum of 26 volumes may be assigned. By appending the drivers to FLEX with the SQUEEZE command and adding a short boot to my ROM, I boot directly from the Winchester. All this was done by following the extensive documentation which is included.

I suggest that you print the .DOC files to get the complete description. Keep up the good work.

Sam Green
8693 Old Town Drive
University City, MO 63132

*Editor's Note: Our thanks to Doctor Green and Leo Taylor for the following. They have been a couple of very knowledgeable contributors to the pages of 68 Micro Journal for many years. We appreciate their sharing their knowledge and programming "goodies" with all of us. Thanks Doc and Leo! We need good solid hardware and software articles like these.*

*There is no reason in the world why our S50 bus systems can't be upgraded to state-of-the-art systems. The 6809 has a lot of kick left. What I wonder is, "do we?" How about some more stuff of this quality from some of you other readers, please write or call for our writers guide, or even simpler, just go by the guide-lines on the content page of this issue. Remember, without your support we are nothing. Your contributions are what keeps 68 Micro Journal going!*

*DMW*

## WINCHESTER DRIVE SYSTEM SOFTWARE

By: Leo Taylor

**OVERVIEW**

The Winchester System Software (called WINSYS) is a series of utilities for handling Winchester disk drives using the FLEX or STARDOS operating systems. WINSYS consists of the following:

WINDRV - the disk drivers WINFMT - the disk formatter WINASN - assigns Winchester volumes WINPARK - parks the head over landing zone WINBOOT - bootstrap loader for ROM monitor WINTABLE - user system equates WIN.DOC - this document WINEXAM.DOC - examples for this document

This group of programs provides several features. The system supports maximums of four drives, 1024 tracks per drive, 26 volumes, 256 tracks per volume, and 10 drive numbers. Each volume can be assigned to any drive number at any time using WINASN. FLEX can be bootstraped from a winchester drive when WINBOOT is added to EPROM. Minor features include proper control of the drive select LED, inter-volume protection, compact driver (about 300 bytes), command line parameters, readable source code, easy driver install.

The software was written to be easy to use, but since it supports many volumes one must be careful not to overwrite a volume. Do not try to format the drive until reading the section on setting up the table.

*!!! IMPORTANT NOTE !!!*

If you edit WINTABLE.LIB you must reassemble BOTH the format and assign programs. If they get out of sync YOU WILL LOSE DATA.

**GLOSSARY**

Drive number - FLEX drive number, usually 0-3. Physical drive - winchester or floppy drive select number, from a switch on the drive, usually 0-3. Logical volume - section of the Winchester designated by a volume letter, usually A-Z. The physical drives can have up to 26 volumes. A drive number can have any volume letter assigned. Drive LED - red indicator on the front of the drive. Maximum track - highest track used for data. Parking track - inside track, slightly higher than maximum track. Error correct - feature of some controllers that fixes data errors. EXAMPLE A - printout obtained from a simple system with one disk drive. Each part is numbered, EXAMPLE A1 is the WINTABLE.LIB listing, A2 is the WIMFMT listing, and EXAMPLE A3 is the WINASN listing. EXAMPLE B - printout obtained from my system with two drives and 24 volume letters.

## HARDWARE SUPPORTED

Since I only know a few FLEX users with hard disks, I do not know the limits of WINSYS. The program should work with any hardware compatible with the following tested hardware:

1. Western Digital 1000-05 and 1002-05 controllers (not SASI). 2. Shugart 604 (5 meg) and Seagate 225 (20 meg) drives. 3. Three chip SS-30 interface (see schematic) and David Graves SS-50 board (68 MJ, Oct 82).

## UNDERSTANDING VOLUME LETTERS

The entire WINSYS is based on the concept of logical volumes which you must understand to use the programs. A logical volume is a section of the hard drive which may be from 2 tracks to 256 tracks. This program assigns a letter to each volume. In EXAMPLE A, letter B is a volume with 70 data tracks starting at track 31. Let's assume we formatted volume B with WINFMT and assigned it to drive number 1 with WINASN. If we catalog this drive (using CAT 1) FLEX will look for a directory on track zero of drive 1. Unknown to FLEX, WINDRV will add 31 to the track number (volume B starts at 31). The directory of volume B will be found at this track, and FLEX will display a catalog of the volume. If we save a file on this drive (using SAVE 1.TEST 100 200) FLEX will want to start the file on track 1 sector 1. WINDRV will add 31 to this track number also, so the file will actually start on track 32.

The important things to realize are:

1. The operation of WINDRV adding offsets to track numbers is transparent to FLEX. No FLEX program can detect the offset to the physical track. 2. Each FLEX drive number can only be assigned one drive letter (track offset) at one time. WINASN is used to change the letter (and track offset) assigned to a drive number. 3. Volumes can be as short as one data track (2 tracks if you count the directory) or as long as 255 data tracks. 4. Since a drive can have many volumes of up to 256 tracks, a drive of up to 1024 tracks (about 65 Megabytes) can be supported under FLEX.

## SETTING UP WINTABLE.LIB

The file WINTABLE.LIB is a library file used by most of the programs in WINSYS. It must be present on the work drive to assemble any of the other programs. It contains all of the user defined system equates, as well as the volume sizes. It MUST be edited before you can begin assembling the programs.

Before you assemble any of the programs, you must consider what size volumes you wish to have. It will take awhile (and perhaps several false starts) to get the sizes the way you need them. The volume sizes can be as small as one data track or as large as 255 data tracks. Though one track seems too small to be of value, bear in mind that with a 4 head drive one track is 128 sectors (over 1/3 of a SA-400 floppy). For example, I use a single track volume called LIBRARY for the 'C' compiler library files.

The first half of WINTABLE.LIB contains several equates pertaining to the hardware. These are explained with comments in the library file, the following is an expanded description of each item.

'DEFALT' is the drive number which will be assigned to the first volume when WINDRV is loaded, usually drive 3 will be volume A. Later, if you are going to boot FLEX from the winchester drive, the default drive must be changed to zero to allow access to STARTUP.TXT.

'DELAY' is used to time out waiting for the controller to indicate ready after indicating busy. This is only used when the controller is not powered up. If your drive is always powered when the computer is on, you can set the delay to 10000 and forget it. Too short of a delay may cause 'DRIVE NOT READY' on the first Winchester access after power up. If you power down your controller when not using your drive, the time out will prevent hanging the system if you access the drive. Repeat: most users should use 10000 and forget it! 'DRV1ST' is the number of the first (or only) physical drive. One would expect this to be drive 0, later I'll explain a small advantage to having the first drive select at 1. 'DRVORG' is the start of the drive table and the disk drivers. This location must be 'out of the way', where FLEX programs will not overwrite the drivers. If you have the interface board by Graves it has a RAM chip to bold the disk drivers. If you have an SWTP MP-09 there is RAM space on the CPU for the drivers. Usually you will want the drivers located in the $E800 or $F000 area. WINDRV only needs about 300 bytes. 'ECFLAG' is a flag used for selecting error correction on the controllers that support it such as WD1002.05. If you have a WD1000-05, then set the flag to $00. If you have a WD1002-05, you will want error correct enabled, set the flag to $80. 'HBASE' is simply the starting address of the interface. 'HEADS' is the number of data heads per drive. For example the SA-604 has 4 heads for its two platters. 'INTERL' is the default interleave factor. You may overide this number by letting WINFMT prompt you for interleave. You will need to experiment with interleave factors to find the optimum number for your system. You can set this to 16 for now. 'MAXDRV' is the largest drive number your FLEX/STARDOS system can handle. Most FLEX systems are limited to 0-3, STARDOS and my FLEX allow drives 0-9. 'NODRIV' is the phantom drive selected to turn off all the drive select LEDs. If you don't have a third drive, use 3. The driver will select this number to turn off all lights. 'OFFSET' is a trick to allow old mother-boards to support eight address ports. If you have the Graves board, or you have at least 8 addresses per port, set this equate to $00. If you have 4 addresses per port (like me) then you can still use the interface by connecting A5 to an unused I/O buss line (such as UD3). This will result in the board decoding 8 addresses, though in two groups of 4. My card responds to $E010-$E013 and $E030-$E033. Setting OFFSET to $1C allows access to this split address. 'PRKTRK' is set to slightly over TRACKS. The drive manufacturers build the drive with extra tracks to provide a 'landing zone' where you can safely park the heads. Most users allow the heads to fall wherever they were at power off, usually on top of a directory! 'PRECMP' is set to 32 for most drives. This track multiplied by four is the start of precompensation. 'SEEKSP' is set to 0 for most drives. See the section SEEK and RESTORE for more information. 'TRACKS' is simply the number of tracks per drive. For example the SA-604 has 160 tracks so TRACKS is 160.

The latter section of WINTABLE.LIB contains 26 volume size equates. The numbers are the size, in DATA tracks, of the volume for each letter. If SIZA is equated to 30 then volume A will have 30 data tracks and 1 directory track (31 total). The directory of volume A will be at track zero and the data tracks will be tracks 1-30. If SIZB is equated to 70 then volume B will have its catalog at track 31, immediately after volume A. The 70 data tracks for volume B will be physical tracks 32-101. The total number of tracks (SIZA+1,SIZB+1,etc) must add up to multiples of the number of tracks per drive. If not, WINSYS will give an error message. In other words, a volume can't extend across two drives. If you have more than one drive, and a volume ends exactly at the end of the first drive, then the next volume will start on the next physical drive. You needn't use all the tracks on a drive, WINSYS will not complain if the end of the last volume is less than TRACKS. You may skip any volume letters by equating them to a size of zero.

## WINTABLE.LIB EXAMPLES

All this should become clearer when you look at the examples. EXAMPLE A1 is from a simple system, and is the file that is distributed as WINTABLE.LIB. The first half should be easy to follow. The controller is a WD1000-05 connected to a Graves board, so DRVORG, ECFLAG, OFFSET, and HBASE are set accordingly. The drive is a 5 MEG ST-506 compatible, so TRACKS, PRKTRK, and HEADS are set accordingly. There is only one drive which is jumpered to be selected as zero, so DRV1ST and NODRIV are set accordingly.

The size table is set up with only three volumes. They are selected to total 160 TRACKS. Note that the sum must include the directory tracks:

SIZA SIZB SIZC MAX 30+1 + 70+1 +57+1 = 160

EXAMPLE B1 is from a more elaborate system, mine! The hardware equates are set up for 2 drives (jumpered as 1 and 2). The driver is addressed in a small space in my version of FLEX. The error correct feature of the controller is enabled. The maximum drive number on my system is 9, so that I can have 10 Winchester volumes enabled at once (or 4 floppies, 5 Winny volumes, and a RAMDISK). All but 2 volume letters are used, note that I have added the volume names and drive numbers as comments. Sometimes I could match the letter to the name (S=SOURCE, P=PICTURE). I have duplicate volumes on both drives because my main purpose for the second drive is quick backups.

## FORMATTING DISKS

We're getting down to business! Now that you have set up the tables you can assemble WINFMT. There are no options to edit in WINFMT, it gets the user defined equates from the library file. WINFMT is self contained, you do not need to install the drivers before formatting. After assembly, enter the command WINFMT (do not enter any command line parameters for now). With luck you will see a display like EXAMPLE A2. You may get an error message on the bottom of the display if WINFMT does not like your table. There will be 3 items reported for each volume letter. The size is the same number you entered in the equate table. WINFMT has computed the drive number (always zero in EXAMPLE A2) and the starting track number for the volume. If there is an error in your table (such as a volume extending past TRACKS) the bad volume will be marked with asterisks. All volumes following the error will be marked as bad, since they are computed relative to the bad volume. WINFMT will abort if there is an error, you must fix the table and reassemble before continuing.

When the program is happy with the table, it will prompt for the volume letter to be formatted. The first time through, start with volume A. Note that any volume can be formatted without disturbing the others. The next prompt is for the number of surfaces. You should enter the number of surfaces supported by your drive, though you may enter less for testing purposes. Some drives use one surface for internal servo information, do not try to format that surface! If you enter too high a number, WINFMT will run VERY slow since it is getting errors on the non-existent head. Interleave factor is the next prompt. enter 16 for now. Interleave is discussed in detail later.

The program will now attempt to read the old volume ID, if none exists there will be a pause of a couple of seconds. If the volume has been formatted before, the old name and number will be reported. This is to reduce the chance you will format a volume you need; you can exit WINFMT by pressing return. You will now be prompted for the volume name. This has the same limitations as a FLEX file name: 8 characters, first must be alpha, hyphen and underline allowed, 3 character extension allowed. The last prompt will be for volume number which can be up to 65535.

WINFMT will now proceed with the formatting. There will be a pause as the drive is restored. Empty sectors will be created on the disk, then the sectors are filled with zeroes and the FLEX link. The program will display track numbers as it proceeds to let you know how it is progressing. The numbers should count quickly, if the tracks take over 2 seconds each there is something wrong; check the number of heads, or the head data from the 20 pin cable. Unlike a floppy, when you change the drive select jumper you must also move the head data cable to a different jack on the controller. The speed of the linking phase will vary depending on the interleave, but please do not judge the interleave factor by the format speed! The interleave should be optimized for reading files. If all goes well the volume will be formatted. Try formatting the volume again to see if the volume name and number is reported! At this point you can format the other volume letters, though you may format them several times before the day is over! You might format some of the other volumes using WINFMT with a command line parameter. To format volume C without having to answer several prompts, you can type WINFMT C. The number of heads and the interleave factor will be taken from the library file to save you some typing.

## INSTALLING THE DRIVERS

Assemble WINDRV next. The drivers file is set up as a command file, you load the drivers by calling them as a command. Later, you may want to append them to FLEX; for now you can call them manually or add WINDRV to your STARTUP file. WINDRV will load into two sections of memory. The driver itself will be located at the address DRVORG found in the library file. There will also be a one time initialization program loaded at $C100 which does its thing then jumps to the FLEX cold start. The program will save the old FLEX driver jump table (usually the jumps to the floppy routines) then install its own jump table. Unlike the Zeff-Graves program you do not need to change your winchester driver when you change your floppy driver. Also, you can load another driver after this one such as my RAMDISK.

If all is going well you should be able to catalog the volume A that you formatted earlier (using CAT 3). WINDRV assigns the first letter to drive 3, since there will always be a volume starting at track zero. Unless you equated SIZA to zero, CAT 3 should display a catalog of volume A. It is assumed you will want to change this drive 3 = volume A pair with WINASN after the driver is loaded. I included this default assignment for test purposes so that you can test the drivers without WINASN.

## ASSIGNING WINCHESTER VOLUMES

The next program to assemble is WINASN. Again, no changes need to be made to this file. This program is used to select a volume letter to be used with a drive number. After loading the drivers you could only access volume A, which was assigned to drive 3. Enter the command WINASN without any parameters. You should get a list like EXAMPLE A3. This varies from the WINFMT report in that it displays the volume name (and extension if you entered one). This display tells you what volumes are available, sort of a directory of directories. I won't go so far as to say "UNIX like", but a small step in that direction. Along the bottom of the display is a list of drive numbers with their assigned volume letters. 0=- means no volume assigned, 3=A means volume A is assigned to drive 3. WINASN prompts for drive number/volume letter pairs. If you want to assign drive 1 to volume B enter 1B as an answer to the prompt. Now if you enter CAT 1 you will catalog winchester volume B.

Several questions may be popping up in your mind... What happened to my floppy disk at drive 1? The floppy temporarily vanishes from access by FLEX. All references to drive 1 will access the winchester volume assigned to drive 1. If you want the floppy back, you can enter WINASN 1-. Remember I mentioned 0=- is WINASN's way of indicating no volume assigned? The dash (hyphen) is also used to deassign a volume. Does that mean I can assign all my drive numbers as winchesters? Sure does! Try the command WINASN 1A 2B 3C. Assuming you formatted the first three volume letters, you now have three winchester volumes assigned (for now you don't want to assign a winchester drive as the system drive). Now try CAT 1 2 3 and see what appears. What does WINASN accept as parameters? The program wants (and insists) on number/letter pairs. There can be several pairs on one command line. Commas and spaces can be inserted if you find that more readable. For example WINASN 0 A 2,B 3C,4D5E is perfectly acceptable. Since I have 4 floppies my STARTUP file includes WINASN 4C5G6L7H8W3-, the latter two characters re-enable my floppy at drive 3. What other special characters does WINASN have? Like WINFMT the asterisk (star) is used to mark a volume whose size must be corrected. A pound sign (pig pen) is used to indicate a drive not responding when WINASN tries to read the volume ID. How can I remember the special characters? By remembering another special character of course! I think you all know the familiar plus sign, if used as a parameter you will get a help message. Enter WINASN + to get some help on how to use this command. Must the letters be in any order? No. WINASN 1C,0B,2A,3C will work fine though there is little purpose to use the letter C twice. How does this relate to the FLEX ASN command? The two perform separate functions, but work together as a team. WINASN 3C:ASN W=3 will make volume C the work drive. How long will it take to read all those volume names? I find it takes about 2 seconds for WINASN to list my 24 volumes scattered around two 5 megabyte drives. Until all volumes have been formatted, the command will take longer.

Since this is the command you will use the most, I'd suggest playing with it for awhile. It's really very intuitive since you simply tell it the drive number followed by the volume you want to assigned to that drive.

## PARKING A WINCHESTER

Most people simply turn off their computers when done with them. Unfortunately winchesters do not have any hardware that holds the flying head away from the platter. The head just lowers down as the disk slows, finally coming to a scraping stop! Thousands of disk users do this every day, and little is said about the subject. In my profession of servicing computers I've changed my share of winny drive mechanisms, often as the customer mumbles about how long it's been since the disk was backed up! I don't know if any of the failures were due to not parking, but I'd rather not find out the hard way. When a winny fails, it usually goes all at once without any warning. The makers of the drives have always provided landing zones, a section of the disk that is not used for data. They recommend resting the heads there, especially if the drive is to be moved. I've noticed IBM is now providing parking support for their PC-AT, H-P does likewise for their VECTRA. Perhaps PC-XT users have learned the hard way...

WINPARK is the final program for you to assemble. You may use it if you like, ignore it if you must. There is one option in the source: a flag to enable reporting of the drive and track being parked. I use the command before powering the system down.

***To Be Continued Next Month with SEEK and RESTORE, SELECTING INTERLEAVE FACTOR, BOOTING FLEX FROM WINCHESTER and THE LITTLE THINGS THAT COUNT.***

# Bit-Bucket

*By: All of us*

*"Contribute Nothing - Expect Nothing"*, DMW '86

## MICROWARE UPDATE

The following information was extracted from various Microware publications and are presented for your information.

### OS-9 VERSION 2.2

Microware has released a new 2.2 Version of the Industrial, Personal and Professional OS-9 Operating System. The new version includes an update of selected operating system modules, plus an enhanced version of Microware's C Compiler. This release does not require the re-assembly or modification of any system-level software modules or application programs. Most OEM's have compeled integration and testing of V2.2 and are currently shipping this new version of OS-9.

The module updates includes additional functionality to support three exciting new Microware software products (previewed in the Fall 1987 edition of Pipelines): C Source Level Debugger, Ethernet TCP/IP package and Memory Protection Module (SPU). Of course, the release contains corrections for errors reported in the 2.1 Version of OS-9/680x0.

New features added to C include bit-fields, enumerated data types and the ability to return structures and unions. The use of bit fields can reduce program memory requirements by allowing data to be stored in a more efficient sized variable. Enumerated data types have also been included. Their use makes programs more understandable and error free by permitting constants to be referenced by a symbolic name. And finally, the ability to return structures and unions is useful for manipulation of variables. With the addition of these three new features, Microware's Version 3.0 C Compiler now represents an even more powerful C Language tool for software development.

All changes included in the new OS-9 Version 2.2 are documented fully in Release Notes that accompany the update disks. End uses may refer to these notes for a more exact description of changes made in the operating system. Anyone having questions regarding V2.2 should contact their Microware representative or their system supplier for additional information.

### NEW DEVELOPMENT PAKS FROM MICROWARE MVME 133A, 134 &135 NOW AVAILABLE

New Development Paks for Motorola VME 133A, 134 and 135 CPU's are now available from Microware. The MVME 133A, 134 and 135 are popular 68020 CPU's designed for software development and industrial applications. The new Development Paks have been customized to support both European (MVME 319) and American (MVME 320) configurations. An optional MVME 050 system controller can be added to provide two additional serial ports and one parallel port in addition to other features.

The MVME 133A includes complete device driver or trap handler support for the on-board MC68881 FPU, RS-232 serial ports and 20 Mhz clock. These features make this micro-computer a popular choice for numerically intensive applications. The MVME 134 has a MC68851 Paged Memory Management Unit supported by Microware's OS-9/SSM Security Pak making it an ideal engine for large multi-user systems. The significant feature of the MVME 135 is a VSB private memory bus which OS-9 supports for multi-processor systems and systems incorporating high-speed DMA controllers. As with all Microware/Motorola Development Paks, these versions of OS-9 can support up to 16 megabytes of RAM. Microware has also recently released an Optional MVME Driver package for all Motorola VME Development Paks. The Optional Driver package includes object code for the MVME 335 serial/parallel device drivers and the MVME 350 magnetic tape device drivers.

### MICROWARE WELCOMES ITALIAN AUTHORIZED IMPLEMENTORS

Microware and Eximar S.r.l. of Rome, Italy, have signed an Authorized Implementors agreement for OS-9 technical sales and service in Italy. As Authorized Implementors, Eximar will be working with Mieroware and its Distributors to promote Microware products throughout Italy. Their primary focus will be to provide a complete menu of technical assistance including education, software support, integration and installation services.

Eximar's resident, Carlo Narcisi, previously served in the Italian armed forces and represented Motorola GEG for 15 years prior to forming Eximar in 1983. Eximar's Technical Services manager, Ercole Maccioni, also worked for Motorola Microsystems prior to joining the firm in 1984. Ercole and Robeno Evangelista came to Microware for technical training and product marketing meetings the week of April 18th.

Eximar services the Italian defense electronics industry on behalf of selected American high-technology companies. They also provide installation and integration services for commercial customers in the process control and board-level markets.

Readers interested in more information may contact Eximar at the following address:

*Eximar S.r.l.*
*Via Dei Ciceri, 59 • 00175 -Rome, Italy*
*Phone: (6) 762221 • Telex: 623358*

### ADA CROSS COMPILER NOW AVAILABLE FOR OS-9

Microware announces the availability of the Systeam Ada Cross Compiler for VAX/VMS systems from our West German distributor, Dr. Rudolf Keil, GmbH. The Systeam Ada conforms to the Department of Defense 4.4 Ada specification. Microware now can offer its customers a tool to pursue U. S. government contracts which specify software projects to be developed under the Ada programming language.

The Ada Cross Compiler is hosted on VAX/VMS and produces object code for a Motorola MC68020 CPU and MC68881 FPU running under OS-9. It translates full Ada as specified in the 1983 Ada Reference Manual.

The user interface is east to use. The cross compiler is well integrsted into the underlying host operating system. It can recognize about 500 different English language error messages. Messages are self-explanatory and accurately positioned.

The cross compiler can translate up to 1800 lines per minute on a VAX 8500 and requires 1.5 MBytes of hard disk storage space. The amount of virtual memory required for a given application depends on the size and number of compilation units to be compiled and typically varies between 4 and 7 MBytes.

The cross compiler is delivered on magnetic tape or floppy disk and full documentation is supplied with every cross compiler. To order the Systeam Ada Cross Compiler, or for more information, contact Dr. Keil at the address listed below:

*Dr. Rudolf Keil, GmbH*
*Gerhart-Hauptmann-Str. 30*
*6915 Dossenheim, West Germany*
*Phone: 06221-862091*
*Telex: 461166*
*FAX: 8221-861954*

DMW

## GENERAL MICRO SYSTEMS
INCORPORATED

For further information:
Scott Bowman (714)625-5475

**VME, 32-BIT, 25 MHz CPU BOARD SUPPORTS MASTER/ SLAVE VSB TRANSFERS**

Montclair, Calif., June 7, 1988 – A 32-bit, 25 MHz, CPU board that allows VSB master/slave transfers between all CPUs in a system, thus eliminating the VMEbus bottleneck, is now available from General Micro Systems, Inc.

The GMSV07-VSB-1 is a 68020-based board with a 68881 co-processor for full 32-bit operation, up to 1 M Byte of zero-wait-state dual-ported SRAM on board, two multiprotocol serial ports and a configuration controller with timers. Location monitor/mailbox interrupt techniques support real-time processing.

A SAM™ (Special Application Module) mezzanine module provides a unique VSB bus that actually is a second true system bus. It allows every CPU to be master/slave, looking into every other CPU in the system. It also allows memory card access through the P2 connector. Essentially it creates a broadcast bus that allows any address in the system to respond without handshake, while the processor is running. Since this is all done off the VMEbus it eliminates a traditional bottleneck in VME systems.

Determination as to whether any CPU in the system is master or slave is determined by method of address, not by a control register. Dynamic configuration allows configuration changes during operation. The VSB modules also can perform block transfers through a 32-bit DMA port.

Any bus-traffic intensive application using multiple CPUs in a system can benefit greatly from this ability to transfer data via a parallel data patch.

The GMSV07-VSB-1 can both originate and service interrupts on the VMEbus. Location monitor/mailbox interrupts support real-time multi-processing. A unique Bus Master Boot requires only one boot PROM for an entire multiple CPU system.

**APPLE MACINTOSH™ USERS**

**Save over a $1,000.00 on PostScript Laser Printers! Faster - Finer Quality than the original Apple LaserWriter!**

**New & Demos**

**Cartridges-new-rebuilts -colors-**

**In Chattanooga Call: 615 842-4600**
**QMS-Authorized**

**Data-Comp Division**
"A Decade of Quality Service"
Systems World-Wide
Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

## MICRONICS
RESEARCH CORP.

Dear Don,

Another update to keep you on top of typos, etc, in 'Logically Speaking', all of them, with the exception of those relating to Diagrams 42 and 43, being mine.

MILE 08

p26 [illegible], should be [illegible], not [illegible]
p27 dia 27b, needs arrow-head at Box-2, on line from Box-5
p36 near top, Headers 16,8,4,2,1 slightly out of line. Move to left.
    centre "... [illegible] of the flow-table ..." Delete one "the".
p37 final sentence, Change "The" to "the".
p41 para 2, Change "od" to "of"
    para 4, last line should read "... if you do any work in this ..."
        Delete "done".

MILE 09

Diagram 36a Row-4, should read "5 67", not "5 87"
p17 para 2, line 2, should read "4, 5, 6 and 8 to 2" not "4, 5 and 6 to 2"
p18 para 1, lines 2 and 3, Change "[illegible]" to "[illegible]"
p21 diagram 41, right-hand column, final two entries should have a '0'
p21 diagram 42, all binary numbers should be 5 bits long. Therefore in column 1, the final entry and the one two lines above should have 2 short dashes instead of one long dash. Also items 2, 3 and 6 of this column. Similarly in the final entry and the one two rows above in column 2.
p21 penultimate line, Change "ther's" to "there's".
p22 diagram 43, first two lines should have 2 short dashes instead of the final long dash in the binary numbers, and the final binary number should have 3 short dashes instead of one long one.
p22 diagram 44, 2 dashes instead of one long one in row-2, and the final dash of rows 5 and 6.

Readers need only be advised of item 2 in Mile 08, as typos are relatively insignificant, but they'll need to be aware of ALL of Mile 09's problems. Sorry about those!

Don Williams,
68 Micro Journal,
5900 Cassandra Smith Road,
Hixson, TN 37343

Sincerely,
[illegible]
R. Jones
President

## Classifieds
As Submitted - No Guarantees

MUSTANG-020 20Mhz with 68881. OS9 Professional Package & C $3500. Call Tom (615)842-4600.

AT&T 7300 UNIX PC, UNIX V OS, 1MB Memory, 20 MB Hard Disk, 5" Drive, Internal Modem, Mouse. Best Offer Gets It.

S+System with Cabinet, 20 Meg Hard Disk & 8" Disk Drive with DMAF3 Controller Board. 1-X12 Terminal $4800.

HARD DISK 10 Megabyte Drive - Seagate Model #412 $275.
3-Dual 8" drive enclosure with power supply. New in box. $125 each.
5-Siemens 8" Disk Drive, $100 each.

Tano Outpost II, 56K, 2 5" DSDD Drives, FLEX, MUMPS, $495.
QUME QVT-102 terminal, like new, amber screen $250. or best offer.

SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Parallel Port, MP-09CPU Card- $900 complete.
**Tom (615) 842-4600 M-F 9AM to 5PM EST**

•••

SWTPC Computers (3), S/09 MP-09B, 256K RAM. Each has 1 Parallel port, 2 serial printer ports, hard/floppy controller board, dual 8" drives. One is 2 mhz with FLEX installed, 4 serial ports; others are 1 mhz with FLEX installed, 2 and 4 serial ports. Two 20 MB hard disks, 1 inoperative. 5 serial terminals. Call (602) 991-4983 Make offer.

•••

WANTED used 6809 software. Alen Gordon, M.D., 160 N.W. 176 St., Miami, FL 33169 (305) 653-8000

•••

SYSTEMS, COMPONENTS & SOFTWARE: 6809 GIMIX, Elektra, 6800 GIMIX, SWTP & SSB, 51/4 DSDD Drives, Terminals, 300-9600 Cassette I/F & Backup Software. TSC, CSC, etc. Flex2/9 Software for GIMIX, Elektra & SWTP. OS9LI for Elektra & Helix. Credit Cards Accepted.
Send SASE for list to George Ingram 26W482 National St., Wheaton, IL 60188 or call 312-653-0360 CSTM-TH 7:30PM - 10PM, S-S 7:30 AM - 10PM

•••

GIMIX Micro-20, 16.67Mhz Version with 16.67Mhz MC68881, 2MB RAM, Professional OS/9 68K Version 2.0 and full "C" compiler with 68020 & 68881 support. $1,500.00 or best offer.
Call (301) 428-8214 Ask for John Bing

**NEW!**

**OmegaSoft Pascal for the 68020/68881**

P20K is a Pascal package that will generate code for all of the 68000 series processors, including the 68881 coprocessor. P20K will run on any 68000 series computer running the OS-9/68000 (Microware) or PDOS (Eyring Research) operating systems with 512K or more free memory.

The base package (P20K-B) includes the Compiler, Relocatable Macro Assembler, Linking Loader, Screen Editor, Pascal Shell, Linkage Creator, Host Debugger, Configuration manager, Installation program, and Patch utility. A new feature in this compiler is the ability to either link in the parts of the runtime needed by the program, or to use trap handlers for runtime access, to share the runtime library between programs. Complete operating system interface is also included using pascal procedures and functions. The host debugger allows debugging at both the Pascal and assembly language levels of programs that run on the host operating system. Price for the base package is $575.

The runtime source code option (P20K-R) is available for $100 and includes source code for the operating system interface routines as well as pascal runtime.

The Utility source option (P20K-S) is available for $275 and includes source code for the Screen Editor, Pascal Shell, Host Debugger, Patch utility, and Configuration manager.

The Target debugger option (P20K-T) is $225 and includes object and source code. This program allows Pascal level and assembly level debugging in a system without operating system, by using a serial link connected to the host computer.

Prices do not include shipping charges. Master-Card and Visa accepted. OmegaSoft is a registered trademark of Certified Software Corporation.

Gespac SA, 3, Chemin des Aulx, CH-1228, Geneva/Plan-les-Ouates, Switzerland
TEL 022-713400, TLX 429989

Elsoft AG, Zelgweg 12, CH-5405 Baden-Dättwil, Switzerland, TEL 056-833377, TLX 828275

RCS Microsystems Ltd., 141 Uxbridge Road, Hampton Hill, Middlesex, England
TEL 01-9792204, TLX 8951470

Byte Studio Borken, Butenwall 14, D-4280 Borken, West Germany.
TEL 02861-2147, TLX 813343

Eltec Elektronik GmbH, Galileo-Galilei-Straße, 6500 Mainz 42, Postfach 65, West Germany
TEL 06131-50031, TLX 4187273

PEP Elektronik Systeme GmbH, Am Klosterwald 4, D-8950 Kaufbeuren, West Germany
TEL 08341-8974, TLX 541233

**CERTIFIED SOFTWARE CORPORATION**

616 CAMINO CABALLO, NIPOMO, CA 93444 USA
TEL (805) 929-1395 TELEX 467013
FAX (805) 929-1395 (MID-8AM)

**FLEX™/SK-DOS™/MS-DOS™**

**Transfer Utilities**

**For 68XXX and CoCo* OS-9™ Systems**

**Now READ - WRITE - DIR - DUMP - EXPLORE**
**FLEX, SK-DOS & MS-DOS Disk**

These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks.

*CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.

CoCo Version: $69.95 68XXX Version $99.95

**S.E. Media** ⟷ **PO Box 849, Hixson,TN 37343**
**615 842-6809** **MC/Visa**

**SUPERIOR SOFTWARE**
**FOR YOUR**
**6809**

| | |
|---|---|
| ★ SK•DOS® Disk Operating System | $75.00 |
| ★ Configuration Manual | $50.00 |
| ★ HUMBUG® Monitor | $50.00 |
| ★ MICROBUG Monitor | $30.00 |
| ★ SPELL 'N FIX Spelling Checker | $89.29 |
| ★ STAR-DOS for the Coco | $34.50 |
| ★ CHECK 'N TAX | $50.00 |

**AND 68000**

| | |
|---|---|
| ★ SK•DOS® Operating System | $140.00 |
| ★ HUMBUG® Monitor | $ 50.00 |
| ★ SPELL 'N FIX Spelling Checker (Coming) | |

STAR-K
SOFTWARE SYSTEMS CORPORATION
BOX 209 • MT. KISCO, N.Y. 10549 • 914/241-0287

**TIRED OF SLOW COMPUTERS ??**
**PROGRAMS SEEM TO TAKE FOREVER ??**
**DETROIT DATACOMM OFFERS A SOLUTION !!**

A hardware emulator executing 68000 object code !!

Using a 60Mhz clock with a floating point multiplier from Bipolar Intergrated Technology the system is capable of up to 5 MIPS.
- By running EXISTING software there is NO need to redo or convert programs
- SWAP AND GO remove your existing CPU, plug in the hardware emulator and watch your program run in record time
- Performs hardware square roots a real time SAVER !!

Available July 1988

**Detroit Datacom Inc.**
**1404 West 14 Mile Road**
**Madison Heights, MI. 48071**
**1-(313)-524-2868**

## SOFTWARE FOR 680x AND MSDOS

**SUPER SLEUTH DISASSEMBLERS**
**EACH $99-FLEX $101-OS9 $100-UNIFLEX**
OBJECT-ONLY versions: EACH $50-FLEX,OS9,COCO
interactively generate source on disk with labels, include xref, binary editing
specify 6800,1,2,3,5,8,9/6502 version or Z80/8080,5 version
OS9 version also processes FLEX format object file under OS9
COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not Z80/8080,5) only
68010 disassembler $100-FLEX,OS9,UNIFLEX,MSDOS,UNIX,SKDOS

**CROSS-ASSEMBLERS WITH MACRO CAPABILITIES**
**EACH $50-FLEX,OS9,UNIFLEX,MSDOS,UNIX,SKDOS 3/$100 ALL/$200**
specify: 180x,6502,6801/11,6804,6805,6809,Z8,Z80,8048,8051,8085,68010,32000
modular cross-assemblers in C, with load/unload utilities
sources for additional $50 each, $100 for 3, $300 for all

**DEBUGGING SIMULATORS FOR POPULAR 8-BIT MICROPROCESSORS**
**EACH $75-FLEX $100-OS9 $80-UNIFLEX**
OBJECT-ONLY versions: EAC $50-COCO FLEX,COCO OS9
interactively simulate processors, include disassembly formatting, binary editing
specify for 6800/1, (14)6805, 6502, 6809 OS9, Z80 FLEX

**ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809**

| | |
|---|---|
| 6502 to 6809 | $75-FLEX $85-OS9 $80-UNIFLEX |
| 6800/1 to 6809 & 6809 to position-ind. | $50-FLEX $75-OS9 $60-UNIFLEX |

**FULL-SCREEN XBASIC PROGRAMS** with cursor control
**AVAILABLE FOR FLEX, UNIFLEX, AND MSDOS**

| | |
|---|---|
| DISPLAY GENERATOR/DOCUMENTOR | $50 w/source, $25 without |
| MAILING LIST SYSTEM | $100 w/source, $50 without |
| INVENTORY WITH MRP | $100 w/source, $50 without |
| TABULA RASA SPREADSHEET | $100 w/source, $50 without |

**DISK AND XBASIC UTILITY PROGRAM LIBRARY**
**$50-FLEX $30-UNIFLEX/MSDOS**
edit disk sectors, sort directory, maintain master catalog, do disk sorts, resequence some or all of BASIC program, xref BASIC program, etc.
non-FLEX versions include sort and resequencer only

**CMODEM TELECOMMUNICATIONS PROGRAM**
**$100-FLEX,OS9,UNIFLEX,MS-DOS,UNIX,SKDOS**
OBJECT-ONLY versions: EACH $50
menu-driven with terminal mode, file transfer, MODEM7, XON-XOFF, etc.
for COCO and non-COCO; drives internal COCO modem port up to 2400 Baud

## DISKETTES & SERVICES

**5.25" DISKETTES**
**EACH 10-PACK $7.50-SSSD/SSDD/DSDD**
American-made, guaranteed 100% quality, with Tyvek jackets, hub rings, and labels

**ADDITIONAL SERVICES FOR THE COMPUTING COMMUNITY**
**CUSTOMIZED PROGRAMMING**
we will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications.

**CONTRACT PROGRAMMING**
we will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 680x0, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.

**CONSULTING**
we offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses.

**Computer Systems Consultants, Inc.**
**1454 Latta Lane, Conyers, GA 30207**
**Telephone 404-483-4570 or 1717**

**We take orders at any time, but plan long discussions after 6, if possible.**

**Contact us about catalog, dealer, discounts, and services.**
**Most programs in source: give computer, OS, disk size.**
**25% off multiple purchases of same program on one order.**
**VISA and MASTER CARD accepted; US funds only, please.**
**Add GA sales tax (if in GA) and 5% shipping.**
(UNI)FLEX tm Technical Systems Consultants; OS/9 Microware;
COCO Tandy; MSDOS Microsoft; SKDOS Star-K Software

**Clearbrook Software Group** (604)853-9118

CSG IMS

**CSG IMS** is *THE* full featured relational database manager for OS9/OSK. The comprehensive structured application language and B+Tree index structures make **CSG IMS** the ideal tool for file-intensive applications.

| | |
|---|---|
| CSG IMS for CoCo2/3 OS9 L1/2 (single user) | $169.95 |
| CSG IMS for OS9 L2 or 68000(multi user) | $495.00 |
| CSG IMS demo with manual | $30 |

**MSF - MSDos File Manager for CoCo 3/OS9 Level 2**

allows you to use MSDos disks directly under OS9.

Requires CoCo 3, OS9 L2, SDISK3 driver $45.00

**SERINA - System Mode Debugger for OS9 L2**

allows you to trace execution of any system module, set break points, assemble and disassemble code and examine and change memory.

Requires CoCo3 or Gimix II, OS9 L2 & 80 col. terminal $139.00

**ERINA - Symbolic User Mode Debugger for OS9**

lets you find bugs by displaying the machine state and instuctions being executed. Set break points, change memory, assemble and disassemble code.

Requires 80 column display, OS9 L1/2 $69.00

Shipping: N. America - $5, Overseas - $10

Clearbrook Software Group P.O. Box 8000-499, Sumas, WA 98295

OS9 is a trademark of Microware Systems Corp., MSDos is a trademark of Microsoft Corp.

**SPECIAL**

**ATARI™**

**&**

**OS-9™**

**NOW!**
**If you have either the Atari 520 or 1040 - you can take advantage of the *"bargain of a lifetime"* OS-9 68K and BASIC all for the low, low price of:**

**$150.00**

**Call or Write**

S.E. Media
5900 Cassandra Smith Rd.
Hixson, TN 37343
615 842-4601

## *ATARI & AMIGA CALL*

As most of you know, we are very sensitive to your wishes, as concerns the contents of these pages. One of the things that many of you have repeatedly written or called about is coverage for the **Atari & Amiga™** series of 68000 computers.

Actually we haven't been too keen on those systems due to a lack of serious software. They were mainly expensive"game-toy"systems. However, recently we are seeing more and more honest-to-goodness serious software for the Atari & Amiga machines. That makes a difference. I feel that we are ready to start some serious looking into a section for the Atari & Amiga computers. Especially so since OS-9 is now running on the Atari (review copy on the way for evaluation and report to you) and rumored for the Amiga. Many of you are doing all kinds of interesting things on these systems. By sharing we all benefit.

**This I must stress - *Input from you on the Atari & Amiga.* As most of you are aware, we are a "contributor supported" magazine. That means that YOU have to do your part. Which is the way it has been for over 10 years. We need articles, technical, reviews of hardware and software, programming (all languages) and the many other facets of support that we have pursued for these many years. Also I will need several to volunteer to do regular columns on the Atari & Amiga systems. Without constant input we can't make it fly! So, if you do your part, we certainly will do ours. How about it, drop me a line or give me a phone call and I will get additional information right back to you. We need your input and support if this is to succeed!**

**DMW**

# THE 6800-6809 BOOKS

## ..HEAR YE.....HEAR

## OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles
OS9 USER NOTES

**Information for the BEGINNER to the PRO, Regular or CoCo OS9**

**Using OS9**
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

**Program interfacing to OS9**
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

**Programming Languages**
Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

**Disks Include**
**No typing all the Source Listings in.** Source Code and, where applicable, assembled or compiled Operating Programs. The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing your OWN more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

**BOOK $9.95**

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
Deluxe Binder - - - - - - - - - - - $5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

## FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the TEXT files included in the book and on diskette.

All TEXT files in the book are on the disks.

| | |
|---|---|
| LOGO.C1 | File load program to offset memory — ASM PIC |
| MEMOVE.C1 | Memory move program — ASM PIC |
| DUMP.C1 | Printer dump program — uses LOGO — ASM PIC |
| SUBTEST.C1 | Simulation of 6800 code to 6809, show differences — ASM |
| TERMEM.C2 | Modem input to disk (or other port input to disk) — ASM |
| M.C2 | Output a file to modem (or another port) — ASM |
| PRINT.C3 | Parallel (enhanced) printer driver — ASM |
| MODEM.C2 | TTL output to CRT and modem (or other port) — ASM |
| SCIPKG.C1 | Scientific math routines — PASCAL |
| U.C4 | Mini-monitor, disk resident, many useful functions — ASM |
| PRINT.C4 | Parallel printer driver, without PFLAG — ASM |
| SET.C5 | Set printer modes — ASM |
| SETBAS1.C5 | Set printer modes — A-BASIC |

NOTE: .C1, .C2, etc.=Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included is ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

**Shipping & Handling $3.50 per Book, $2.50 per Disk set**
**Foreign Orders Add $4.50 Surface Mail**
**or $7.00 Air Mail**
**If paying by check - Please allow 4-6 weeks delivery**
*** All Currency in U.S. Dollars**

**Continually Updated In 68 Micro Journal Monthly**

VISA

**Computer Publishing Inc.**
**5900 Cassandra Smith Rd.**
**Hixson, TN 37343**

**(615) 842-4601**
**Telex 5106006630**

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware and Motorola
*68' Micro Journal is a trademark of Computer Publishing Inc.

## !!! Subscribe Now !!!
## 68 MICRO JOURNAL

**OK, PLEASE ENTER MY SUBSCRIPTION**

Bill My: Mastercard ☐ VISA ☐

Card #____________________ Exp. Date________

For 1 Year ____ 2 Years ____ 3 Years ____

Enclosed: $____________

**Name** ______________________________

**Street** ______________________________

**City** ________________ **State** ____ **Zip** ________

**Country** ______________________________

**My Computer Is:** ________________

### Subscription Rates

**U.S.A.: 1 Year $24.50, 2 Years $42.50, 3 Years $64.50**

***Foreign Surface: Add $12.00 per Year to USA Price.**

***Foreign Airmail: Add $48.00 per Year to USA Price.**

***Canada & Mexico: Add $9.50 per Year to USA Price.**

***U.S. Currency Cash or Check Drawn on a USA Bank !**

**68 Micro Journal**
5900 Cassandra Smith Rd.
POB 849
Hixson, TN 37343

**Telephone 615 842-4600**
**Telex 510 600-6630**

## Reader Service Disks

| | |
|---|---|
| Disk- 1 | Filesort, Minicat, Minicopy,Minifms, **Lifetime, **Poetry, **Foodlist, **Diet. |
| Disk- 2 | Diskedit w/ inst.& fixes. Prime, *Prmod, **Snoopy, **Football, **Hexpawn,**Lifetime. |
| Disk- 3 | Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave. |
| Disk- 4 | Mailing Program, *Finddat, *Change, *Testdisk. |
| Disk- 5 | *DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING. |
| Disk- 6 | **Purchase Order, Index (Disk file indx). |
| Disk- 7 | Linking Loader, Rload, Harkness. |
| Disk- 8 | Crtest, Lanpher (May 82). |
| Disk- 9 | Datecopy, Diskfix9 (Aug 82). |
| Disk-10 | Home Accounting (July 82). |
| Disk-11 | Dissembler (June 84). |
| Disk-12 | Modem68 (May 84). |
| Disk-13 | *Initmf68, Testmf68,*Cleanup,*Dskalign,Help,Date.Txt. |
| Disk-14 | *Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib. |
| Disk-15 | Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo). |
| Disk-16 | Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc. |
| Disk-17 | Match Utility, RATBAS, A Basic Preprocessor. |
| Disk-18 | Parse.Mod, Size.Cmd (Sept. 85 Armstrong),CMDCODE, CMD.Txt (Sept. 85 Spray). |
| Disk-19 | Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc. |
| Disk-20 | UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, LS.C, FDUMP.C. |
| Disk-21 | Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more. |
| Disk-22 | Read CPM & Non-FLEX Disks. Fraser May 1984. |
| Disk-23 | ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven, Language Recognition Utility, Anderson March1986. |
| Disk-24 | 68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current. |
| Disk-25 | KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk. |
| Disk-26 | Compacta UniBoard review, code & diagram, Burlison March '86. |
| Disk-27 | ROTABIT.TXT, SUMSTEST.TXT, CONDATA.TXT, BADMEN.TXT. |
| Disk-28 | CT-82 Emulator, bit mapped. |
| Disk-29 | **Star Trek |
| Disk-30 | Simple Winchester, Dec.'86 Green. |
| Disk-31 | *** Read/Write MS/PC-DOS (SK*DOS) |
| Disk-32 | Heir-UNIX Type upgrade - 68MJ 2/87 |
| Disk-33 | Build the GT-4 Terminal - 68MJ 11/87 Condon. |
| Disk-34 | FLEX 6809 Diagnostics, Disk Drive Test, ROM Test, RAM Test - 68MJ 4/88Korpi. |

NOTE:
This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC
*** Denotes 68000 - 6809 no indicator.

**8" disk $19.50**
**5" disk $16.95**

Shipping & Handling -U.S.A. Add: - $3.50
Overseas add: $4.50 Surface - $7.00 Airmail

**68 MICRO JOURNAL**
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4600 - Telex 510 600-6630

# K-BASIC™

**The Only 6809 BASIC to Binary Compiler for OS-9**
**FLEX or SK*DOS**
**Even runs on the 68XXX SK*DOS Systems***

*Hundreds Sold at Suggested Retail:*

~~$199.00~~

• 6809 - OS-9™ users can now transfer their FLEX™ Extended BASIC (XBASIC) source files to OS-9, compile with the OS-9 version and run them as any other OS-9 binary "CMD" program. Much faster than BASIC programs.

• 6809 - FLEX users can compiler their BASIC source files to a regular FLEX ".CMD" file. Much faster execution.

• 68XXX - SK*DOS™ users running on 68XXX systems (such as the Mustang-08/A) can continue to execute their 6809 FLEX BASIC and compiled programs while getting things ported over to the 68XXX. SK*DOS allows 6809 programs to run in emulation mode. This is the only system we know of that will run both 6809 & 68XXX binary files.

K-BASIC is a true compiler. Compiling BASIC 6809 programs to binary command type programs. The savings in RAM needed and the increased speed of binary execution makes this a must for the serious user. And the price is now RIGHT!

**Don't get caught up in the *"Learn a New Language"* syndrome - Write Your Program in BASIC, Debug it in BASIC and Then Compile it to a .CMD Binary File.**

**For a LIMITED time save over 65%... This sale will not be repeated after it's over! ***

**SALE SPECIAL:**
**$69.95**

## SPECIAL
## Thank-You-Sale

*Only From:*

**S.E. Media™**

CPI

5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

A Division of Computer Publishing Inc.
Over 1,200 Titles - 6800-6809-68000

* K-BASIC will run under 68XXX SK*DOS in emulation mode for the 6809.
Price subject to change without notice.

## PT-68000 SINGLE BOARD COMPUTER

The PT68K2 is Available in a Variety of Formats
From Basic Kits to Completely Assembled Systems

**BASIC KIT (8 MHZ)** - Board, 68000, HUMBUG MONITOR + BASIC in ROM, 4K STATIC RAM, 2 SERIAL PORTS, all Components **$200**

**PACKAGE DEAL** - Complete Kit with Board 68000 10 MHZ, SK*DOS, 512K RAM, and all Necessary Parts **$564**

**ASSEMBLED BOARD** (12 MHZ) Completely Tested, 1024K RAM, FLOPPY CONTROLLER, PIA, SK*DOS **$899**

**ASSEMBLED SYSTEM** - 10 MHZ BOARD, CABINET POWER SUPPLY, MONITOR + KEYBOARD, 80 TRACK FLOPPY DRIVE, CABLES **$1299**
For A 20 MEG DRIVE, CONTROLLER and CABLES Add **$345**

**PROFESSIONAL OS9** **$500**

**FEATURES**

- MC68000 Processor, 8 MHZ Clock (optional 10,12.5 MHZ)
- 512K or 1024K of DRAM (no wait states)
- 4K of SPAM (6116)
- 32K,64K or 128K of EPROM
- Four RS-232 Serial Ports
- Floppy disk controller will control up to four 5 1/4", 40 or 80 track.
- Clock with on-board battery.
- 2 - 8 bit Parallel Ports
- Board can be mounted in an IBM type PC/XT cabinet and has a power connector to match the IBM type power supply.
- Expansion ports - 6 IBM PC/XT compatible I/O ports. The HUMBUG™ monitor supports monochrome and/or color adaptor cards and Western Digital winchester interface cards.

**PERIPHERAL TECHNOLOGY**
**1480 Terrell Mill Rd., Suite 870**
**Marietta, Georgia 30067**
404/984-0742
VISA/MASTERCARD/CHECK/C.O.D.

Send For Catalogue
For Complete Information On All Products

™SK*DOS is a Trademark of STAR-K SOFTWARE SYSTEMS CORP.
™OS9 is a Trademark of Microware

## *DATA-COMP* SPECIAL

### Heavy Duty Power Supplies

**For A limited time our HEAVY DUTY SWITCHING POWER SUPPLY. These are BRAND NEW units. Note that these prices are less than 1/4 the normal price for these high quality units.**

Make: Borchert
Size: 10.5 x 5 x 2.5 inches
Including heavy mounting bracket and heatsink

Rating: in 110/220 volts ac (strap change) Out: 130 watts

Output: +5v - 10 amps
+12v - 4.0 amps
+12v - 2.0 amps
-12v - 0.5 amps

Mating Connector: Terminal strip
Load Reaction: Automatic short circuit recovery
**SPECIAL: $59.95 each**
**2 or more $49.95 each**
Add: $7.50 each S/H

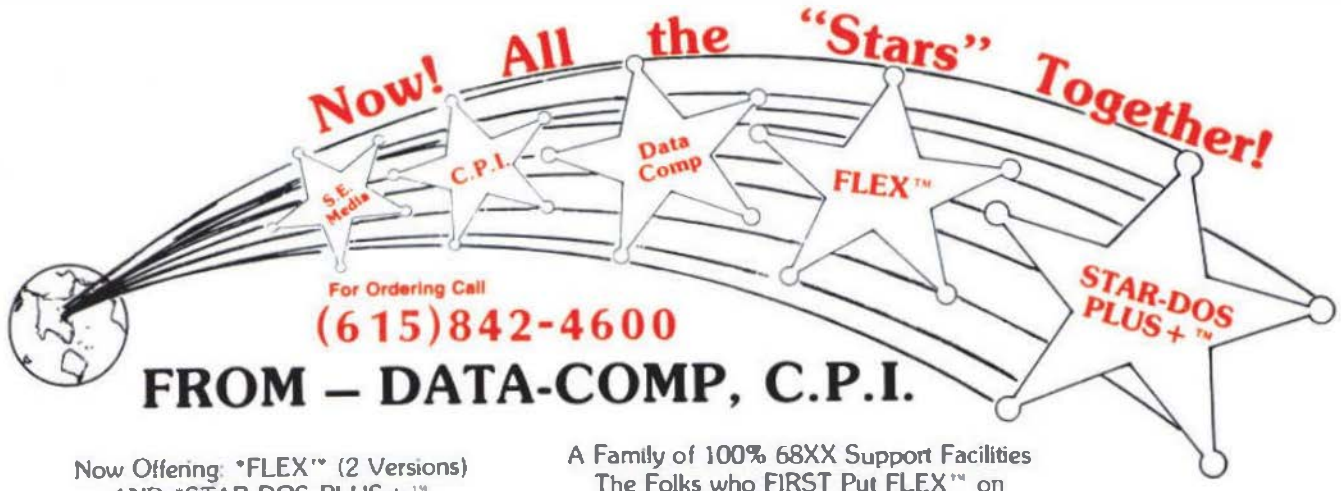
Make: Borchert
Size: 10.75 x 6.2 x 2.25 inches

Rating: 110/220 ac (strap change) Out: 81 watts

Outputs: +5v - 8.0 amps
+12v - 2.4 amps
+12v - 2.4 amps
+12v - 2.1 amps
-12v - 0.4 amps

Mating Connectors: Molex
Load Reaction: Automatic short circuit recovery
**SPECIAL: $49.95 each**
**2 or more $39.95 each**
Add: $7.50 S/H each

**5900 Cassandra Smith Rd., Hixson, Tn. 37343 Telephone 615 842-4600 Telex 510 600-5630**



For Ordering Call

**(615)842-4600**

**FROM – DATA-COMP, C.P.I.**

Now Offering: *FLEX™ (2 Versions) AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on The CoCo

**FLEX-CoCo Sr.**
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

**STAR-DOS PLUS+**
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.95

**FLEX-CoCo Jr.**
without TSC Editor & Assembler
$49.95

**PLUS**

**ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE**

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

**TSC Editor**
Reg $50.00
**NOW $35.00**

**TSC Assembler**
Reg $50.00
**NOW $35.00**

**CoCo Disk Drive Systems**

| | |
|---|---|
| 2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES SYSTEM WITH POWER SUPPLY,CABINET,DISK DRIVE CABLE,J&M NEW DISK CONTROLLER JFD-CP WITH J-DOS,RS-DOS OPERATING SYSTEMS. | $469.95 |

* Specify What CONTROLLER You Want J&M, or RADIO SHACK

| | |
|---|---|
| THINLINE DOUBLE SIDED DOUBLE DENSITY 40 TRACKS | $129.95 |

**Verbatim Diskettes**

| | |
|---|---|
| Single Sided Double Density | $ 24.00 |
| Double Sided Double Density | $ 24.00 |

**Controllers**

| | |
|---|---|
| J&M JFD-CP WITH J-DOS | $139.95 |
| WITH J-DOS, RS-DOS | $159.95 |
| RADIO SHACK 1.1 | $134.95 |
| RADIO SHACK Disk CONTROLLER 1.1 | $134.95 |

**Disk Drive Cables**

| | |
|---|---|
| Cable for One Drive | $ 19.95 |
| Cable for Two Drives | $ 24.95 |

**MISC**

| | |
|---|---|
| 64K UPGRADE FOR C,D,E,F, AND COCO II | $ 29.95 |
| RADIO SHACK BASIC 1.2 | $ 24.95 |
| RADIO SHACK DISK BASIC 1.1 | $ 24.95 |
| DISK DRIVE CABINET FOR A SINGLE DRIVE | $ 49.95 |
| DISK DRIVE CABINET FOR TWO THINLINE DRIVES | $ 69.95 |

**PRINTERS**

| | |
|---|---|
| EPSON LX-80 | $289.95 |
| EPSON MX-70 | $125.95 |
| EPSON MX-100 | $495.95 |

**ACCESSORIES FOR EPSON**

| | |
|---|---|
| 8148 2K SERIAL BOARD | $ 89.95 |
| 8149 32K EXPAND TO 128K | $169.95 |
| EPSON MX-RX-80 RIBBONS | $ 7.95 |
| EPSON LX-80 RIBBONS | $ 5.95 |
| TRACTOR UNITS FOR LX-80 | $ 39.95 |

CABLES & OTHER INTERFACES CALL FOR PRICING

**DATA-COMP**
5900 Cassandra Smith Rd.
Hixson, TN 37343

**SHIPPING**
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

**(615)842-4600**
For Ordering
Telex 5106006630

*An Ace of a System in Spades! The New*

# MUSTANG-08/A™

**Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the $500.00 OS-9 C compiler! This offer won't last forever!**

## *NOT* 128K, *NOT* 512K
## *FULL* 768K No Wait RAM

The MUSTANG-08™ system took every hand from all other 68008 systems we tested, running OS-9 68K!

The MUSTANG-08 includes OS9-68K™ and/or Peter Stark's SK*DOS™. SK*DOS is a single user, single tasking system that takes up where *FLEX™ left off. SK*DOS is actually a 68XXX FLEX type system *(Not a TSC product.)*

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It is a speed *whiz on disk I/O*. Fact is: the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that is just a small part of the story! See benchmarks.

SALE!

System includes OS-9 68K or SK*DOS - Your Choice

Specifications:

| | | |
|---|---|---|
| CPU | MC68008 | 12 Mhz |
| RAM | 768K | 256K Chips |
| | No Wait States | |
| PORTS | 4 - RS232 | MC68681 DUART |
| | 2 - 8 bit Parallel | MC6821 PIA |
| CLOCK | MK48T02 | Real Time Clock Bat. B/U |
| EPROM | 16K, 32K or 64K | Selectable |
| FLOPPY | WD1772 | 5 1/4 Drives |
| HARD DISK | Interface Port | WD1002 Board |

**Now more serial ports - faster CPU Battery B/U - and $850.00 OS-9 Professional with C compiler included!**

***$400.00** See Mustang-02 Ad - page 5 for trade-in details

Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68030

MUSTANG-08

LOOK

```
                                 Seconds  32 bit   Register
                                          Integer  Long
Other      68008  8   Mhz  OS-9   68K...18.0...9.0
MUSTANG-08   10   Mhz  OS-9   68K....9.8...6.3
```

C Benchmark Loop

```
Main()
{
        /* int i; */
        register long i;
        for (i=0; i < 999999; ++i);
}
```

Now even faster!
with 12 Mhz CPU

| C Compile times: OS-9 68K Hard Disk | |
|---|---|
| MUSTANG-08 8 Mhz CPU | 0 min - 32 sec |
| Other popular 68008 system | 1 min - 05 sec |
| MUSTANG-020 | 0 min - 21 sec |

## 25 Megabyte Hard Disk System

## $2,398.90

**Complete with PROFESSIONAL OS-9 includes the $500.00 C compiler, PC style cabinet, heavy duty power supply, 5" DDDS 80 track floppy, 25 MegByte Hard Disk - Ready to Run**

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be easily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (system must have 128K in addition to its other requirements). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC    MUSTANG-08 is a trademark of CPI

**Data-Comp Division**

A Decade of Quality Service"

Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

* Those with SWTPC hi-density FLEX 5" - Call for special info.